ALGORITHMIC ASPECTS OF SPARSE APPROXIMATIONS

THÈSE N^{O} 3936 (2007)

PRÉSENTÉE LE 19 OCTOBRE 2007

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE TRAITEMENT DES SIGNAUX 2

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Philippe JOST

ingénieur en systèmes de communication diplômé EPF
de nationalité suisse et originaire de Langnau Im Emmental (BE)

acceptée sur proposition du jury:

Prof. M. Hasler, président du jury
Prof. P. Vandergheynst, directeur de thèse
Dr L. Daudet, rapporteur
Dr C. De Vleeschouwer, rapporteur
Prof. J.-Ph. Thiran, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2007

Table of contents

Abstract

Typical tasks in signal processing may be done in simpler ways or more efficiently if the signals to analyze are represented in a proper way. This thesis deals with some algorithmic problems related to signal approximation, more precisely, in the novel field of sparse approximation using redundant dictionaries of functions.

Orthogonal bases permit to approximate signals by just taking the N waveforms whose associated projections have maximal amplitudes. This nice property is no longer valid if the used base is redundant. In fact, finding the best decomposition becomes a NP Hard problem in the general case. Thus, suboptimal heuristics have been developed; the best known ones are Matching Pursuit and Basis Pursuit. Both remain highly complex which prevent them from being used in practice in many situations. The first part of the thesis is concerned with this computational bottleneck. We propose to create a tree structure endowing the dictionary and grouping similar atoms in the same branches. An approximation algorithm, called Tree-Based Pursuit, exploiting this structure is presented. It considerably lowers the cost of finding good approximations with redundant dictionaries.

The quality of the representation does not only depend on the approximation algorithm but also on the dictionary used. One of the main advantage of these techniques is that the atoms can be tailored to match the features present in the signal. It might happen that some knowledge about the class of signals to approximate directly leads to the dictionary. For most natural signals, however, the underlying structures are not clearly known and may be obfuscated. Learning dictionaries based on examples is an alternative to manual design and is gaining in interest. Most natural signals exhibit behaviors invariant to translations in space or in time. Thus, we propose an algorithm to learn redundant dictionaries under the translation invariance constraint. In the case of images, the proposed solution is able to recover atoms similar to Gabor functions, line edge detectors and curved edge detectors. The two first categories were already observed and the third one completes the range of natural features and is a major contribution of this algorithm.

Sparsity is used to define the efficiency of approximation algorithms as well as to characterize *good* dictionaries. It directly comes from the fact that these techniques aim at approximating signals with few significant terms. This property was successfully exploited as a dimension reduction method for different signal processing tasks as analysis, de-noising or compression. In the last chapter, we tackle the problem of finding the nearest neighbor

to a query signal in a set of signals that have a sparse representation. We take advantage of sparsity to approximate quickly the distance between the query and all elements of the database. In this way, we are able to prune recursively all elements that do not match the query, while providing bounds on the true distance. Validation of this technique on synthetic and real data sets confirms that it could be very well suited to process queries over large databases of compressed signals, avoiding most of the burden of decoding.

Keywords

sparse approximation,	redundant dictionaries
dictionary learning,	data mining
fast algorithms,	nearest neighbor
structured dictionaries,	greedy algorithms

Version abrégée

Certaines opérations courantes dans le domaine du traitement de signal peuvent être réalisées de manière plus simple ou plus précise si les signaux à traiter sont représentés de manière adéquate. Cette thèse étudie quelques problèmes algorithmiques liés à l'approximation des signaux ; plus précisément nous nous sommes intéressés à la représentation parcimonieuse de signaux en utilisant des dictionnaires redondants.

Une base orthogonale permet d'approximer simplement un signal en prenant les N éléments dont les projections sur le signal ont la plus grande amplitude. Cette propriété n'est pas valide dans le cas de bases redondantes, pour lesquelles, trouver la meilleure approximation est un problème très complexe, NP-Complet dans le cas général. Des solutions sous-optimales tentant de résoudre ce problème existent. Matching Pursuit et Basis Pursuit en sont les plus fameux représentants. Malgré tout, ces algorithmes restent très coûteux en puissance de calcul et, de ce fait, sont difficilement utilisables en pratique. La première partie de cette thèse traite de ce coût prohibitif. Nous proposons de réorganiser le dictionnaire redondant en groupant les fonctions de base (atomes) similaires. Cette manière de faire aboutit à la création d'une structure en arbre. Un algorithme d'approximation, que nous avons nommé Tree-Based Pursuit, utilise cet arbre pour décomposer des signaux. Cela permet de diminuer considérablement la complexité de calcul nécessaire pour trouver de bonnes approximations en utilisant des dictionnaires redondants.

La qualité d'une approximation ne dépend pas uniquement de l'algorithme utilisé pour trouver la décomposition mais aussi du dictionnaire. Un des avantages majeurs de cette famille de techniques est que les atomes composant le dictionnaire peuvent être taillés sur mesure en fonction des signaux à traiter. Il peut arriver que, connaissant les phénomènes physiques générant les signaux, il soit possible de trouver facilement un très bon dictionnaire. Pour la majorité des signaux naturels, les structures sous-jacentes ne sont pas connues ou sont cachées. Dans cette situation, l'apprentissage automatique de dictionnaire à partir de données est une alternative viable au design manuel de dictionnaires. La majorité des signaux naturels ont un comportement invariant par translation dans le temps ou dans l'espace. Nous proposons une solution qui intègre cela et apprend des dictionnaires invariants par translation. Appliqué à des images naturelles, notre algorithme retrouve des fonctions similaires à des atomes de Gabor, des détecteurs de lignes ainsi que des détecteurs de contours. Si les deux premières catégories ont déjà été observées, la troisième s'ajoute

à la collection des attributs des images naturelles et est une des contributions majeures de cet algorithme.

La parcimonie permet de quantifier l'efficacité d'un algorithme d'approximation aussi bien que de mesurer la qualité d'un dictionnaire redondant. En effet, le but ultime est d'approximer un signal avec peu de termes très significatifs. Cette propriété a été utilisée avec succès pour différentes opérations dans le domaine du traitement de signal comme l'analyse, le dé-bruitage ou encore la compression de données. Dans le dernier chapitre, nous nous attaquons au problème du plus proche voisin. Etant donné un signal requête, nous recherchons celui qui en est le plus proche dans un grand ensemble. Les signaux considérés ont une représentation parcimonieuse de qualité que nous allons utiliser pour approximer de manière efficace et rapide la distance entre la requête et les signaux de la base de données. Nous sommes capables d'éliminer rapidement ceux qui sont trop différents de ce que nous cherchons. Cette manière de faire est validée expérimentalement avec des données synthétiques ainsi que sur des données réelles. Cela confirme que cet algorithme peut être utilisé pour traiter des requêtes sur des bases de données de signaux compressés, évitant ainsi les coûts liés au décodage.

Mots Clés

représentations parcimonieuse, dictionnaires redondants
apprentissage de dictionnaires, data mining
algorithmes rapides, plus proche voisin
dictionnaires structurés, algorithmes gloutons

List of figures

Dissertation

1 Introduction

1.1 Motivations

An approximation is an inexact representation that is good enough to be useful. Working with approximations instead of original signals may be of great interest in many situations. Handling the original data may be computationally too complex or an approximation leads to results that are sufficiently good given the target application e.g. using 3.14 instead of the real value of π. These statements are also valid in the field of signal processing especially when dealing with high dimensional signals.

Typical signals are images, video sequences, sounds, biomedical signals as ECG or EEG. Signal Processing consists in manipulating and analyzing signals. Among the most famous treatments, one can cite compression, source separation, denoising, feature extraction, and many others. Some of these tasks may benefit from a clever representation of the information at hand. Representing signals with basic building blocks that essentially synthesize the information is one of the major theme in approximation theory. The quality of such representations is generally measured by the number of basis elements needed to achieve a good approximation: the less the better. This concept is referred to as sparsity.

It is known since the early successes of wavelet analysis that sparse expansions very often result in efficient algorithms for typical tasks in signal processing. An interesting and increasingly popular way of achieving sparsity is to turn to very redundant systems i.e. approximate signals as weighted sums of functions, called atoms, from a collection of unit energy functions spanning the signal space, called dictionary. Redundancy implies that distinct atoms may be highly correlated. Approximation using redundant dictionaries often allows for short-length representation of signals, since the probability of finding a sparse approximation generally increases with the redundancy of the dictionary. A common way to generate redundant dictionaries is to take a small set of basis functions that may be trans-

lated at any position in the signal. This family of redundant dictionaries generated from a small set of waveforms that are submitted to different transforms are called structured dictionaries.

Redundancy in the dictionary implies that a decomposition is not unique; additionally finding the sparsest one is a NP Hard problem. Suboptimal algorithms that are sufficiently good for most applications have been proposed. They get rid of some constraints of the original problem. However, they still remain computationally expensive. Lowering the complexity is a challenge that may lead these techniques from the academic world to real applications.

Redundant dictionaries based techniques provide flexibility for representing data. This statement holds only if the dictionary is adapted to the class of signals to treat. In numerous cases, the physical processes generating the signals is known and represent an important piece of information for designing efficient dictionaries. For most natural signals this is not the case. Additionally, these possibly high dimensional signals often exhibit complex behaviors and may contain noise. The field concerned with revealing hidden structures from data is called data mining and is gaining in interest since mid nineties. However, researchers concerned with redundant expansions did not get interested in these techniques until very recently. Dictionary learning is a promising technique that permits to dream of very sparse representations by identifying the fundamental structures from the data.

The aim of this thesis is to propose efficient algorithmic solutions for sparse representation of signals using redundant dictionaries. We tackle the problem of finding good representations at reasonable costs by structuring redundant dictionaries in a smart way that can be exploited by an approximation algorithm. A second objective of this work is to develop a dictionary learning algorithm. Again, we seek for a low complexity method. This goal is achieved by proposing a solution that does not explicitly include a sparsity constraint which is the costliest part of most learning algorithms. Finally, we propose to exploit the sparsity to propose an efficient solution to the nearest neighbor problem in high dimensional spaces.

1.2 Roadmap and Main Contributions

Sparsity as well as redundancy imply that most problems in the field of sparse approximation do not have a unique solution. In this thesis, we focus on three topics in this context and for all the treated problems, we propose algorithms that are efficient from a computational point of view. Each chapter begins with an introductory part that guides the reader trough the main results and methods proposed in the literature.

Chapter 2 deals with finding sparse representations. It begins with an introduction to the sparse approximation problem and presents the *classical* algorithms, such as Matching Pursuit and Basis Pursuit. They are representative of two families of algorithmic solutions developed in order to find sub-optimal solutions to the original NP Hard problem. Redundancy in the dictionary implicitly introduces redundant computations to be done for the approximation algorithms. Starting from this observation, section 2.3 proposes to

group similar atoms together and to represent the so created clusters by a centroid, called molecule. This clustering inspired methodology eventually leads to a tree when applied recursively. Atoms from the initial dictionary are leaves of the tree. Section 2.4 presents an algorithm, called Tree-Based Pursuit, taking advantage of the previously defined structure. Finding the best atom in a signal corresponds to choosing the best path in the tree. The two following sections validate the proposed method. First, in section 2.5 we prove that under certain conditions, the algorithm systematically finds atoms belonging to the good part of the tree. Second, section 2.6 presents experiments that validate the proposed methodology. In addition to these experiments, section 2.7 presents an application using Tree-Based Pursuit. The idea is to create a very low bit rate coder for human faces. This application in the field of biometric data handling is becoming more and more popular. Our coding scheme is compared JPEG 2000 and shows great improvements.

Chapter 3 deals with dictionary learning. The introduction, section 3.2, presents the state of the art methods in this field. These methods generally involve a sparsity model that is either stochastic or algorithmic dependant. Section 3.3 proposes a dictionary learning algorithm called MoTIF, for Matching of Time Invariant Filters, that does not explicitly use a sparsity model. MoTIF iteratively searches for waveforms that represent well features present in the signals whilst being as decorrelated as possible from the previously found ones. The underlying assumption is that dictionaries made of meaningful waveforms will mandatorily lead to sparse representations of the data. Additionally, MoTIF aims to find translation invariant waveforms i.e. it learns structured dictionaries. The behavior of the algorithm is illustrated by learnt dictionaries for natural images. Two experiments present the abilities of the proposed solution for dictionary learning. First, signals are generated by randomly adding waveforms coming from a small set of functions. MoTIF is used to recover the used waveforms. Second, a dictionary has been learnt from an audio signal. This dictionary is compared to an allround dictionary made of Gabor atoms in terms of approximation abilities. Different researchers used MoTIF or were inspired by it. Section 3.5 presents their main results. In particular, MoTIF has been used to learn audio-visual features in the field of multi-modal signal processing.

Chapter 4 is application oriented. We propose a solution for finding those signals in a database that are closest to a query. The signals of interest have a sparse approximation over a basis or a redundant dictionary. They are referred to as *compressible signals*; images, music and speech signals belong to this class. Section 4.3 presents how sparsity can be used to approximate quickly the distances between the query and all elements of the database. We present an algorithm that uses this principle to prune recursively all elements that do not match the query while providing bounds on the true distance. Section 4.4 uses a probabilistic approach to speed up the algorithm at the cost of introducing uncertainty on the final results. The experimental part, section 4.5, validates this technique on synthetic and real data sets and confirms that it could be very well suited to process queries over large databases of transformed signals, avoiding most of the burden of decoding.

Chapter 5 is a discussion of the obtained results. It aims at drawing some conclusions as well as present possible extensions and future research directions to this work.

2 Tree-Based Pursuit

2.1 Motivations

Finding the best linear expansion using a redundant dictionary of functions is a daunting task and even a NP-Hard problem [19] in the general case. Despite the difficulty to find the *best*, sparsest solution, it is possible to find sufficiently *good* representations that are nearly optimal. Sub-optimal heuristics have been developed that recover the main components of a function in a redundant dictionary. Among the most popular algorithms that find good suboptimal solutions to the sparsest signal representation problem, we can cite Matching Pursuit [51] and Basis Pursuit [11]: both find a solution by relaxing some constraints of the original optimization problem. For most applications, these solutions are generally close enough to the optimum. Even if specific optimizations are possible for particular classes of dictionaries, the complexity of these algorithms however remains very high in general.

Several methods have been proposed in order to decrease the computational complexity to find sparse signal expansions. They are generally modified versions of Matching Pursuit or Basis Pursuit and propose modifications of either the search algorithm itself, or the dictionary. It is indeed possible to introduce small changes to obtain efficient search algorithms. A two stage design is proposed in [12, 57, 66], where the original dictionary functions are approximated by linear combinations of very simple, elementary vectors. The search is then performed in the space of elementary vectors, hence a great reduction in computational complexity. In [65] a greedy algorithm solving Basis Pursuit is presented. It has the advantage to be much faster than the interior point method traditionally used for Basis Pursuit.

Approximation of functions of the dictionary or special constructions can also lead to efficient search algorithms, without an important penalty on the approximation performance

[57]. Multiscale [34] or subband dictionaries [20] can be used to decrease the search complexity, where the linearity of the inner product can even be further exploited to speed-up the computation, at the price of higher memory requirements. Similarly, [32] proposed to use a dictionary that is based on damped sinusoids, which can be efficiently derived using simple recursive filter banks. Since the size of the dictionary has obviously an important impact on the search complexity, several studies have also been proposed to prune the dictionary to its most meaningful elements, by vector quantization for example [13, 69]. In general, these methods however only apply to specific dictionaries. Few efforts were invested in having an efficient implementation of Matching Pursuit until the release of the Matching Pursuit Toolkit [35].

In this chapter, we study the reduction of the computational complexity of the search for the sparsest signal expansion, for any highly redundant dictionary. It naturally leads to the notion of data structuring, which becomes critical when the amount of data gets very large. Dictionary functions with similar properties can be clustered together, in order to facilitate the search for the sparsest representation. Clustering is a widely used technique when the amount of data is huge and hides the underlying structures, see [39] for a survey. Clustering algorithms depend on a measure to quantify the similarity between two objects. Proper data arrangement then allows for the development of tree data structures, which can be efficiently used for search when a huge amount of data is present [67]. Tree search has been proposed in [16] in order to improve the performance of Matching Pursuit expansion. We however propose to study tree-based pursuit from a complexity reduction perspective, as an interesting trade-off between efficient implementation and sufficiently sparse signal approximation.

Section 2.2 proposes an overview of linear expansions using redundant dictionaries of functions. Section 2.3 presents a structuring method that allows to represent a subset of highly correlated atoms by a single element, called molecule. Hierarchical clustering then allows for building trees, where each node corresponds to a molecule that encompasses the characteristics of all its relative children. A tree construction method is then proposed that respects the necessary conditions for nodes at each level to be sufficiently incoherent. A tree-based pursuit algorithm is then proposed in Section 2.4, and exploits the tree structure to reduce the computational complexity of the pursuit. Performance and characteristics of the algorithm are analyzed in Section 2.5. A bound is derived, which ensures that molecules cover the same span as the initial dictionary. A minimal condition ensuring that the algorithm chooses only *good* molecules under the root node is also presented. Section 2.6 illustrates the performance of Tree-Based Pursuit in terms of approximation and complexity, compared to Matching Pursuit. A very low bit-rate face coder is presented in section 2.7. It is a practical application that benefits from the gain provided by Tree-Based Pursuit.

2.2 Sparse Approximations

For the last few years, there has been a tremendous activity in the field of sparse approximations. This is partly motivated by the potential of the related techniques for typical tasks

in signal processing such as analysis, dimensionality reduction, de-noising or compression. This section provides an overview of the main recent results on sparse approximations, and practical algorithms.

2.2.1 Formal Problem

Given a d dimensional signal f in a real vector space, the central problem faced is the following: compute a good approximation $\tilde{f}_N$ as a linear superposition of N basic elements, which are often called *atoms*, picked up in a huge collection of signals $\mathcal{D}$, usually referred to as a dictionary. The dictionary is said to be redundant when its cardinality $|\mathcal{D}| \gg d$. The approximant $\tilde{f}_N$ is sparse when $N \ll d$, where

$$\tilde{f}_N = \sum_{k=0}^{N-1} c_k g_{\gamma_k}, \quad g_{\gamma_k} \in \mathcal{D}. \tag{2.1}$$

There is no particular requirement concerning the dictionary, except that it should span the signal space $\mathcal{H}$ and that the atoms are unit-norm signals. A signal can thus be exactly represented as a linear combination of atoms from the dictionary. In addition, there is no prescription on how to compute the coefficients c_k in eq. (2.1). The main advantage of this class of techniques is the complete freedom in designing the dictionary, which can then be efficiently tailored to closely match signal structures.

This problem is better studied under the form of the following constrained optimization :

$$P_0 : \text{minimize } \|c\|_0 \text{ subject to } \|f - \sum_{k=0}^{N-1} c_k g_{\gamma_k}\|_2 \leq \epsilon$$

where $\|c\|_0$ counts the number of nonzero entries in the sequence $\{c_k\}$ and ϵ is the maximal acceptable error.

Usually, finding the solution of P_0 would be a hopeless combinatorial problem. In fact, without adding special constraints to the dictionary or to the signals, the sparse approximation problem is NP-Hard (Corrolary 2.3 of [80]) and finding the best solution is a combinatorial problem. However, for some applications, suboptimal solutions may be sufficient. In order to find such a solution, some constraints of the original constraints have to be *relaxed*. These algorithms can be classified in two categories:

Greedy algorithms The global solution is found iteratively.

Optimization methods The l_0 norm is replaced by another sparsity measure for which the problem is not NP-Hard.

2.2.2 Greedy algorithms

Greedy algorithms iteratively construct an approximant by selecting the element of the dictionary that best matches the signal at each iteration. The pure greedy algorithm is

known as *Matching Pursuit* [51]. Assuming that all atoms in $\mathcal{D}$ have norm one, we initialize the algorithm by setting $R^0 f = f$ and we first decompose the signal as

$$R^0 f = \langle g_{\gamma_0}, R^0 f \rangle g_{\gamma_0} + R^1 f \,,$$

where g_{γ_0} is chosen so as to maximize the correlation with $R^0 f$:

$$g_{\gamma_0} = \arg\max_{\mathcal{D}} |\langle g_{\gamma_0}, R^0 f \rangle| \,.$$

We then iterate the procedure on the residual $R^1 f$ and, after N steps, build the following approximation :

$$f = \sum_{n=0}^{N-1} \langle g_{\gamma_n}, R^n f \rangle g_{\gamma_n} + R^N f \,,$$

where the norm of the residual (approximation error) satisfies

$$\|R^N f\|^2 = \|f\|^2 - \sum_{n=0}^{N-1} |\langle g_{\gamma_n}, R^n f \rangle|^2 \,.$$

The algorithm ends when some stopping criterion is met. For example, one may iterate until the energy of the residual is lower than some threshold or the number of terms of the approximation may be fixed *a-priori*.

The introduction of the Matching Pursuit algorithm in the signal processing community has been inspiring for many researchers and different variations of this algorithm were proposed. The most famous ones are Orthogonal Matching Pursuit [62] and Weak Matching Pursuit [76].

Orthogonal Matching Pursuit uses the same atom selection method as Matching Pursuit. Then, all coefficients of the expansion are recomputed such that the residual is orthogonal to all previously found atoms. Matching Pursuit only guarantees that the residual is orthogonal to the last found atom.

Weak Matching Pursuit uses a different criterion to select the atom. It allows to take atoms whose correlation is not maximal with the residual. It was driven by the fact that the dictionary may be very large and that computing only parts of the possible scalar product lowers the complexity.

The greedy heuristic finds in general a good approximant to the problem in polynomial time; its approximation abilities is deeply studied in [77]. There is however no guarantee on the optimality of the solution, except in the case where sufficient conditions are set on the dictionary [78]. However, polynomial time still does not mean fast! Typical implementations of Matching Pursuit suffer from high computational complexity when compared to most orthogonal transforms. It was long believed that it could prevent these techniques from being used in applications. Recent research shows that this bottleneck is not far from being broken. The MPTK toolkit [35] provides a very fast implementation of Matching Pursuit. The authors systematically analyzed the bottlenecks and surprisingly found out that for highly dimensional signals, finding the atom whose projection is maximal is costly.

They created a tree structure that keeps trace of the maximal values. The search for the maximum is done by a traversal of the tree. When the residual is changed, the scalar products are only computed in the places where changes happened and the tree structure is updated. In addition, they make use of fast transforms to compute groups of inner products at once. Their software is able to achieve 0.25× real time for a typical Matching Pursuit analysis scenario applied to a one hour long audio track.

In the remainder of this chapter, we propose to reduce the complexity by an efficient organization of the dictionary. We propose to group similar atoms together, and represent them by a unique element called *molecule*. Applying clustering recursively on atoms and molecules yields a hierarchical tree structure, that can be exploited to design a search algorithm with greatly reduced complexity.

2.2.3 Optimization methods

The measure of sparsity used in P_0 is the l_0-norm which makes this problem highly non convex. Efficient minimization algorithms can thus not be used to find a solution. The class of algorithms presented in this section relax this constraint by replacing the $l0$-norm by another measures for the sparseness.

Basis Pursuit [11] uses the $l1$-norm. The problem is rewritten as follows:

$$P_1 : \text{minimize } \|c\|_1 \text{ subject to } \|f - \sum_{k=0}^{N-1} c_k g_{\gamma_k}\|_2 < \epsilon$$

It is straight forward to transform P_1 into a Quadratic Programming problem. This algorithm finds jointly a subset of the dictionary and the corresponding weights. The complexity of this method is high which makes it not usable when dealing with high-dimensional signals. Even what is thought to be a small image can be considered as high dimensional in this case.

Like for the greedy algorithms, people have been inspired by the principle of *convex relaxation* and different other methods have been tried to find good approximations. The most famous ones are the Method of Frames [18] and FOCUSS [33].

As for the greedy algorithms, there is no guarantee that the found solution is also the solution of P_0. Generally, optimization methods offer better results than the greedy approaches at the cost of more complex computations. Like for the greedy algorithms, there is hope as efficient implementations of techniques for solving large scale linear or quadratic programs have recently emerged [5].

2.2.4 Dictionary

There are no special conditions on the dictionary. Generally, it is supposed that it should span the signal space. This condition guarantees that any signal could be represented as a linear combination of atoms from the dictionary. It is very intuitive to see that the approximation is tightly linked with the composition of the dictionary. This is true in two

different ways. First, the *quality* of the approximation and second the *performance* of the approximation algorithm.

In order to be able to characterize a dictionary, different measures have been suggested. The most popular measure is the coherence μ defined as follows:

$$\mu = \sup_{\substack{g_i, g_j \in \mathcal{D} \\ i \neq j}} |\langle g_i, g_j \rangle| . \tag{2.2}$$

Despite its simplicity, this measure provides an important piece of information about the dictionary. It tells us how two atoms from the dictionary are at least different. Additionally, it has the nice property to be easily computable.

The so defined coherence has however an important drawback as it does absolutely not take into account the local structures of the dictionary. The cumulative coherence $\mu_1(m)$ (or Babel function) appeared in [78] and in a slightly different form in [23] provides more information about the dictionary. It is defined as follows :

$$\mu_1(m) = \max_{|\Lambda|=m} \max_{i \notin \Lambda} \sum_{j \in \Lambda} | \langle g_i, g_j \rangle | . \tag{2.3}$$

The cumulative coherence measures how a group of atoms is at least different from a fixed atom. For an orthogonal base, $\mu_1(m) = 0$ for all possible values of m. The class of redundant dictionaries for which the cumulative coherence grows slowly are said to be *quasi-incoherent* [80]; they are not too far from being orthogonal bases (although they may be highly overcomplete).

The performance of greedy algorithms like Matching Pursuit are tightly linked to the structure of the dictionary. The coherence μ described above is often not sufficient to represent the properties of a dictionary, since it represents a worst case bound, and does not take into account the local structures of the dictionary. The structural redundancy [31] of a dictionary provides important information about the structure of a redundant dictionary. Matching Pursuit converges exponentially fast in finite dimension [19, 51]. There exist two constants $\alpha > 0$, reflecting the optimality of the pursuit algorithm, and $\beta > 0$, characterizing the redundancy of the dictionary, such that

$$||R^{n+1} f|| \leq (1 - \alpha^2 \beta^2)^{1/2} ||R^n f||, \tag{2.4}$$

where β can be expressed as

$$\beta = \inf_{a, \|a\|=1} \sup_{g_i \in \mathcal{D}} | \langle a, g_i \rangle | . \tag{2.5}$$

This equation confirms that the algorithm will behave well, provided there is always an atom closely aligned with the residual. The structural redundancy β can be seen as a measure of the size of the *biggest* hole in the dictionary. It is upperbounded, $\beta \leq 1$, and it is increasing with the cardinality of the dictionary. The structural redundancy can be computed in simple cases [31]. This characteristic shows that the properties of the signal, dictionary and algorithm, are tightly linked.

2.2.5 Recovery

The algorithms presented previously provide suboptimal solutions regarding the original problem P_0. In some situations, it may be known in advance that a signal to represent admits a sparse representation in a given dictionary. Naturally, a question emerge: under which conditions is it possible to find the solution of P_0 using the suboptimal algorithms?

Trivially, greedy strategies such as Matching Pursuit and Orthogonal Matching Pursuit are able to solve P_0 if the dictionary is an orthogonal base. When using redundant dictionaries, it is also trivial to find examples where Matching Pursuit fails to find a good solution [11].

Simple greedy strategies such as Matching Pursuit and Orthogonal Matching Pursuit are able to recover very good approximants [80]. On the downside, these results only hold for a limited class of dictionaries : $\mathcal{D}$ has to be sufficiently *incoherent*. More properties of such dictionaries can be found in [25, 36, 78]. These results tell us that, if a sufficiently sparse solution exists in a sufficiently incoherent dictionary, it can be found by solving a problem closely connected to P_0. However, in practice, redundant dictionaries do most of the time not satisfy the incoherence condition while still leading to good results.

2.3 Structuring Redundant Dictionaries

2.3.1 From atoms to molecules

This section discusses clustering of a generic, redundant dictionary, which eventually leads to the creation of a tree structure. First, it describes the problem of representing a group of highly correlated dictionary atoms by a unique element. We then discuss the characteristics that are necessary for a dictionary to be efficiently clustered and organized in a tree structure.

Let the elements of the dictionary $\mathcal{D} = \{g_i\}_{i\in\Gamma}$ be labelled by the index set Γ. A sub-dictionary $\mathcal{D}_\Lambda$ is such that $\mathcal{D}_\Lambda = \{g_i\}_{i\in\Lambda}$, where $\Lambda \subset \Gamma$ and $\Lambda \neq \emptyset$. A collection of sub-dictionaries $\{\mathcal{D}_{\Lambda_i}\}$ forms a partition of the dictionary $\mathcal{D}$ if $\bigcup_i \Lambda_i = \Gamma$ and $\Lambda_i \bigcap \Lambda_j = \emptyset$, $\forall i \neq j$. If the atoms in $\mathcal{D}$ are sufficiently uncorrelated, a simple greedy algorithm is able to recover a sparse approximation of the signal (see for example [78]). This is not the case for highly correlated redundant dictionaries. It can be explained intuitively by the fact that high correlation in the dictionary can fool the pursuit and result in wrong choices. We are thus going to try to represent a highly correlated sub-dictionary $\mathcal{D}_{\Lambda_i}$ by a single molecule, while at the same time minimizing the correlation among molecules. This procedure should result in a set of molecules that behaves like a (quasi) incoherent dictionary.

Let us first define the minimal coherence λ_Λ of a sub-dictionary by :

$$\lambda_\Lambda = \min_{i,j\in\Lambda} \mid \langle g_i, g_j \rangle \mid . \tag{2.6}$$

A sub-dictionary will be referred to as *reducible* when λ_Λ is positive and sufficiently big. In order to quantify the adequation of the molecule in representing the atoms in the sub-

dictionary $\{\mathcal{D}_{\Lambda_i}\}$, a distance measure has to be defined. Let $d(g_i, g_j)$ be a measure of the distance between two unit energy atoms g_i and g_j. We chose to use the following distance measure, derived from the simple cosine function :

$$d(g_i, g_j) = 1 - | < g_i, g_j > |^2. \tag{2.7}$$

Note that an atom g_i can be considered as equivalent to $-g_i$, from an approximation point of view, the sign of the weights c_k in eq. (2.1) could be reversed. The distance measure given in eq. (2.7) is independent of the direction of g_i as $d(g_i, -g_i) = 0$.

Most clustering algorithms represent a cluster by a centroid whose mean distance to all elements it represents is minimized. Let us define the optimal centroid or unit norm molecule m_Λ^{opt}, for a sub-dictionary $\mathcal{D}_\Lambda$, by :

$$m_\Lambda^{\text{opt}} = \arg \min_{\substack{m \\ \|m\|=1}} \sum_{i \in \Lambda} d(m, g_i). \tag{2.8}$$

Using the distance measure defined in eq. (2.7), the optimal centroid becomes :

$$\begin{aligned} m_\Lambda^{\text{opt}} &= \arg \min_{\substack{m \\ \|m\|=1}} \sum_{i \in \Lambda} 1 - |\langle m, g_i \rangle|^2, & (2.9) \\ &= \arg \max_{\substack{m \\ \|m\|=1}} \sum_{i \in \Lambda} |\langle m, g_i \rangle|^2, & (2.10) \\ &= \arg \max_{\substack{m \\ \|m\|=1}} m^* A_\Lambda A_\Lambda^* m \,, & (2.11) \end{aligned}$$

where the columns of the matrix A_Λ are the atoms of the sub-dictionary $\mathcal{D}_\Lambda$. The molecule m_Λ^{opt} is the eigenvector associated to the biggest eigenvalue of the matrix $A_\Lambda A_\Lambda^*$. The eigenvalues of $A_\Lambda A_\Lambda^*$ are equal to the eigenvalues of $A_\Lambda^* A_\Lambda$ (see theorem 1.3.20 of [37]). This last matrix is the Grammian of A_Λ. Figure 2.4 illustrates the reduction capabilities of a molecule regarding a group of similar atoms. As the matrix $A_\Lambda A_\Lambda^*$ is symmetric, the associated eigenvalues are real and the associated eigenvectors are orthogonal. The molecule m_Λ^{opt} is also equivalent to the dominant left singular vector of the matrix A_Λ [37].

2.3.2 Dictionary characterization

In the previous section, we introduced the definition of molecule in order to structure the *information* at hand in a highly redundant sub-dictionary. We will now see how a dictionary can be partitioned into disjoint sub-dictionaries represented by molecules through a simple clustering procedure. Further recursive application of clustering on the set of molecules results in a hierarchical tree structure that will be used by the search algorithm.

We previously stated that representing a sub-dictionary by a molecule makes sense only for *reducible* sub-dictionaries. By extension, a dictionary $\mathcal{D}$ is said to be *reducible* if it contains a partition $\{\mathcal{D}_{\Lambda_i}\}$, such that all its sub-dictionaries are reducible and $|\{\mathcal{D}_{\Lambda_i}\}| \ll |\mathcal{D}|$, i.e., the number of sub-dictionaries is much smaller than the number of atoms in the dictionary. A special case of *reducible* dictionaries is represented by the *block incoherent*

dictionaries [64]. These dictionaries are such that it is possible to find a partition having a small block coherence μ_B defined by :

$$\mu_B = \max_{i \neq j} \max_{\substack{k \in \Lambda_i \\ l \in \Lambda_j}} | \langle g_k, g_l \rangle | \ . \tag{2.12}$$

If $\mathcal{D}$ is *reducible*, then the coherence μ of $\mathcal{D}$ is large; the reverse is however not necessarily true. A dictionary $\mathcal{D}$ can have a large coherence μ without being *reducible*, due to the fact that the coherence given in eq. (2.2) only reflects an extreme and local property of the dictionary. Similarly, the quantity β defined in eq. (2.5), or the structural redundancy [31], also reports an extreme property of the dictionary. For *block incoherent* dictionaries, the structural redundancy is low and provides some *inter* sub-dictionaries redundancy measure. It is however closely related to the block-coherence μ_B given in eq. (2.12).

The cumulative coherence is a refinement of the simple coherence measure and therefore provides much more information about the dictionary. It is defined as follows :

$$\mu_1(m) = \max_{|\Lambda|=m} \max_{i \notin \Lambda} \sum_{j \in \Lambda} | \langle g_i, g_j \rangle | \ . \tag{2.13}$$

A dictionary whose cumulative coherence grows slowly (i.e. $\mu_1(m) \ll \mu m$) is said to be *quasi-incoherent* [78]. If it grows fast, it is at least possible to have one highly correlated sub-dictionary. The cumulative coherence can be bounded using the coherence, $\mu_1(m) \leq m\mu$. In the special case of *block incoherent* dictionaries, a better bound on the cumulative coherence $\mu_1(m)$ can even be proposed. Let k be the cardinality of the most populated highly correlated sub-dictionary, we then have :

$$\mu_1(m) \leq \begin{cases} m\mu & \text{if } m < k. \\ (k-1)\mu + (m-k+1)\mu_B & \text{if } m \geq k. \end{cases} \tag{2.14}$$

The cumulative coherence provides more accurate *local* information than the coherence, but is more complex to compute. Moreover, a fast growing cumulative coherence is not a sufficient condition for a dictionary to be *reducible*: it reflects the behavior of the dictionary in the region of the space of signals that is best *covered* by the dictionary [80]. For example, in the case of *block incoherent* dictionaries, the cumulative coherence grows rapidly from $\mu_1(0)$ up to $\mu_1(k-1)$ and then grows slowly, with k being the cardinality of the most populated sub-dictionary. Figure 2.1 presents the evolution of the cumulative coherence for a dictionary having two highly redundant parts. For $m = 5$, there is a sharp inflection of the curve as the cardinality of the most populated group of atoms is $k = 6$. To summarize, a quasi-incoherent dictionary has both small coherence, and small structural redundancy, and its cumulative coherence grows slowly. Block incoherent dictionaries rather have a large coherence and a cumulative coherence that grows fast up to an inflexion point at $m = k - 1$, and then grows slowly. Block incoherent dictionaries are good candidates for one-step clustering of atoms into molecules.

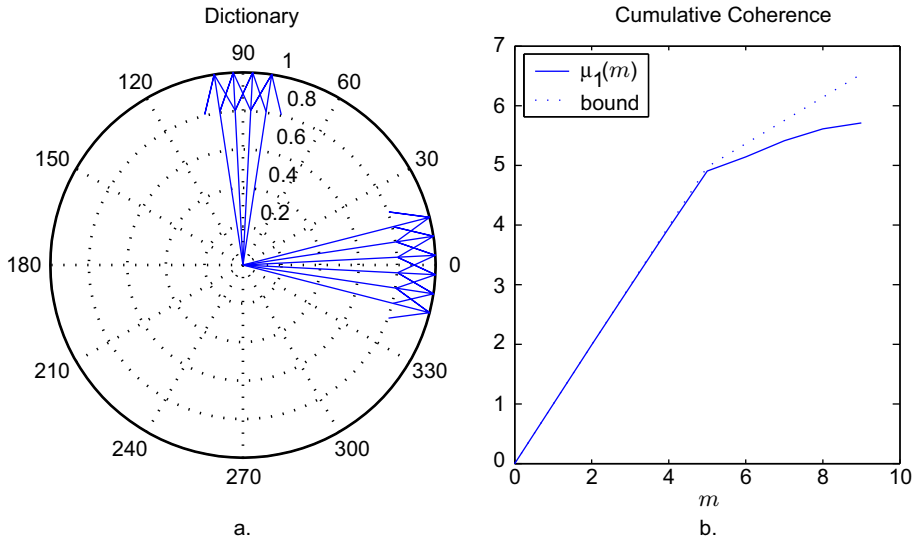


Figure 2.1 – *(a) Simple block incoherent dictionary made of two highly redundant parts. (b) Evolution of its cumulative coherence and the upper bound provided by eq. (2.14).*

2.3.3 Tree-structured dictionaries

The hypothesis that the dictionary is *reducible* ensures that it is possible to partition it into *reducible* sub-dictionaries, and to recursively find molecules. However, we have not yet provided a way to compute the partition of $\mathcal{D}$ in sub-dictionaries. Our ultimate goal is to have as few sub-dictionaries as possible, with atoms within each sub-dictionary that are as similar (correlated) as possible, and atoms from different sub-dictionaries as different (uncorrelated) as possible. We propose a clustering approach that starts from an existing dictionary and endows it with a tree structure $\mathcal{T}$, with nodes $t_i \in \mathcal{T}$. The sub-dictionaries are seen as clusters of atoms, and the associated molecules are the centroids of each cluster. Each node $t_i = \{a_i, m_i\}$ of the tree is associated to a list a_i containing the indices of its children and to a molecule m_i representing these children through eq. (2.8). A leaf node t_i is associated to an original atom from the dictionary $\mathcal{D}$, and a_i contains the index of that atom in $\mathcal{D}$. The root node of the tree is labeled t_0 and has no associated molecule. See Figure 2.6 for an illustration of these notations.

In general, two different clustering approaches can be chosen: (i) a top-down approach that tries to divide the *reducible* dictionary (or sub-dictionary) into sub-dictionaries, that satisfy the similarity constraints, and (ii) a bottom-up approach that groups similar atoms/molecules together as long as similarity constraints are satisfied. A top-down approach using constraints on similarity has been introduced in [21] and is called *diametrical clustering*. This algorithm was developed for gene clustering to fit an observation stating that genes with anti-correlated expression patterns can be functionally similar. The same observation is true for a dictionary approach of signal decomposition : two anti-correlated atoms have the same behavior as they capture the same structure.

In this thesis, we will follow a bottom-up approach, which consists in grouping nodes,

starting from atoms, to create new nodes and molecules. The bottom-up approach is better appropriate to the clustering of arbitrary dictionaries, since the number of clusters does not need to be known in advance. The bottom-up approach presented here sets the cardinality k of each cluster. Algorithm 2.3.3 presents the method. Initially, it creates nodes containing the atoms from a dictionary $\mathcal{D}$. The recursive part consists in finding groups $\{G\}$ of nodes that can be merged. A set Ω_G of molecules is associated to G. We propose a *weak* and a *strong* rule. The *weak* version defines $\Omega_G = \{m_i\}_{i \in G}$, while for the *strong* decision rule, Ω_G contains the molecules associated to the leaf nodes that are the descendants of the different nodes of G (i.e., $\Omega_G \subset \mathcal{D}$). The set of nodes G is merged if $\max_{\substack{i,j \in G \\ i \neq j}} d(m_i, m_j) < \delta$, where δ is an *a priori* fixed threshold.

Algorithm 1 Tree Creation by grouping.

INPUT: $\mathcal{D} = \{g_i\}$, the dictionary.
k, the cardinality of the clusters.
δ, the grouping threshold.
OUTPUT: $\mathcal{T} = \{t_i\}$, the tree.
ALGORITHM:
$a_i = \emptyset$, $\forall\ 0 < i \leq |\mathcal{D}| + 1$, leaf nodes have no children.
$m_i = g_{i-1}$, $\forall\ 0 < i \leq |\mathcal{D}| + 1$, molecules at leaf nodes.
$t_i = \{a_i, m_i\}$, leaf nodes of the tree.
$\mathcal{T} = \{t_i\}$, initial tree.
$L = \{i\}_{i=1}^{|\mathcal{D}|+1}$, list of candidates for grouping.
$G = \arg\max_{G \subset L, |G|=k} \lambda_{\Omega_G}$, subset of nodes to group.
while $1 - \lambda^2_{\Omega_G} < \delta$ **do**
  $m_{|\mathcal{T}|+1} = m^{opt}_{\Omega_G}$, create the new molecule.
  $a_{|\mathcal{T}|+1} = G$, list of children.
  $L = L \setminus G$, remove grouped nodes from the list.
  $L = L \cup \{|\mathcal{T}| + 1\}$, add index of new node to the list.
  $t_{|\mathcal{T}|+1} = \{a_{|\mathcal{T}|+1}, m_{|\mathcal{T}|+1}\}$, create new node.
  $\mathcal{T} = \mathcal{T} \cup t_{|\mathcal{T}|+1}$, add the new node to the tree.
  $G = \arg\max_{G \subset L, |G|=k} \lambda_{\Omega_G}$, subset of nodes to group.
end while
$m_0 = 0$, no molecule at root node.
$t_0 = \{L, m_0\}$, list of nodes at first level.
$\mathcal{T} = \mathcal{T} \cup t_0$, add root node to the tree.

Finding the best group of k nodes is still a combinatorial problem, but it can be easily solved for small values of k (our results are based on trees created with $k = 2$). The tree can be constructed off-line, without penalizing the pursuit algorithm. Figure 2.2 illustrates the construction of a binary tree, for a dictionary of 12 random vectors. The most similar atoms are paired together, until the algorithm reaches level 1 with 3 molecules, which are too incoherent to be further clustered.

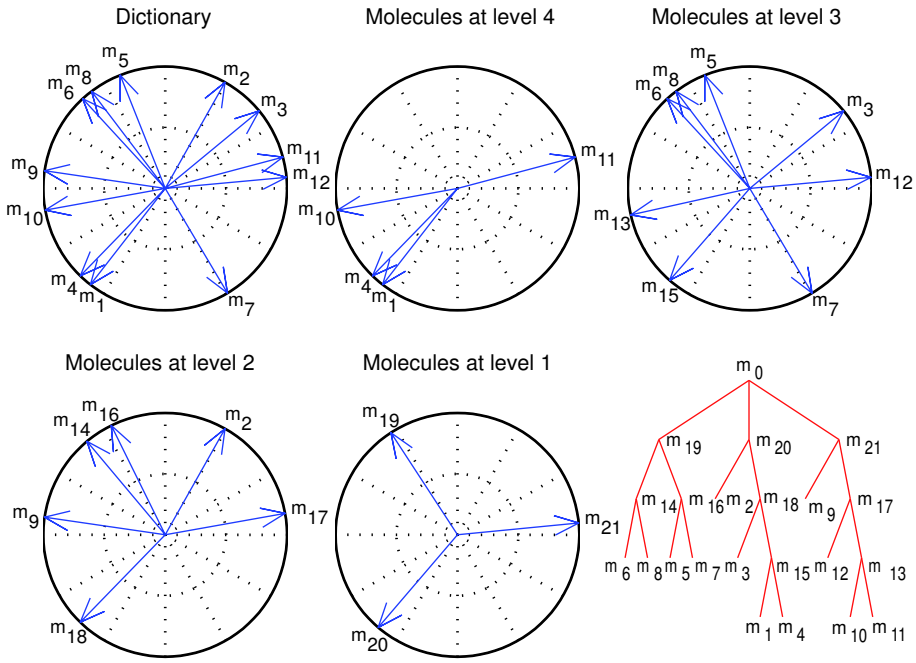


Figure 2.2 – *Creation of a tree on top of a 2D dictionary. The upper-left part shows all atoms in the dictionary. The bottom-right part summarizes the structure of the tree. The other parts correspond to the molecules or atoms present at the different levels of the tree.*

2.4 Tree-Based Pursuit algorithm

2.4.1 Tree-based search

In a sense, a single iteration of Matching Pursuit can be seen as a classification problem where each atom corresponds to a class of signals. Its aim becomes to successively map the residual signal to a class according to a given distance measure. When considering the greedy approximation problem as an iterative classification problem, the tree structure can be used to divide the decision into smaller steps in a manner similar to a decision tree. Matching Pursuit simply tries all possibilities to find the best class. The use of a hierarchical structure allows to discard an important part of the dictionary at each node. In the following, we describe a practical implementation of this technique, the Tree-Based Pursuit algorithm. Like Matching Pursuit, the proposed algorithm iteratively searches for a good atom to approximate a residual signal $R^n f$. Instead of testing all possible atoms from $\mathcal{D}$, Tree-Based Pursuit uses the tree structure $\mathcal{T}$ that groups similar atoms in the same subtree. The search starts at the root node and goes down through the tree until a leaf node is reached. At each node, the algorithm chooses the child whose molecule best approximates the signal (i.e., the one that leads to the highest amplitude of the scalar product with the residual).

In practice, a dictionary $\mathcal{D}$ is often built using several generating functions, which are translated to different positions in the signal space, e.g., in time or space. Let T_p be the

operator that translates a generating function at position p on the support of the atom and leaves the energy unchanged. Tree-Based Pursuit is described by Algorithm 2.4.1 for dictionaries built using generating functions that can be translated at any place on the support of the signal to approximate. However, it remains valid for any kind of dictionary. For dictionaries that do not explicit the translation of atoms, the search of the optimal position is simply discarded.

Algorithm 2 Tree-Based Pursuit algorithm

INPUT: $\mathcal{T} = \{a_i, m_i\}$, the tree structured dictionary
σ, the size of the local search window
f, the signal to approximate.
OUTPUT: $\{g_n\}$, the set of chosen atoms
$\{c_n\}$, the set of corresponding projections.
INITIALIZATION: $R^0 f = f$, $n = 0$
repeat
$[p_{opt}, a_{opt}] = \arg\max_{p,a \in a_0} |\langle R^n f, T_p m_a \rangle|$
while $|a_{opt}| > 1$ **do**
$[p_{opt}, a_{opt}] = \arg\max_{p,a \in a_{opt}} |\langle R^n f, T_p m_a \rangle|$, $|p - p_{opt}| < \sigma$
end while
$g_{n+1} = T_{p_{opt}} g_{a_{opt}}$
$c_{n+1} =< R^n f | g_{n+1} >$
$R^{n+1} = R^n f - c_{n+1} g_n$
$n = n + 1$
until Stop condition is met.

At the root node, the scalar products between the residual $R^n f$ and the molecules of the nodes at the first level of the tree are computed. This step is equivalent to Matching Pursuit using the molecules of the first tree level as dictionary. When using dictionaries built using generating functions, the initial step also gives the position of the best molecule. It can be considered as an energy localization phase. Note that in our case, this localization method is particularly efficient, since molecules really represent the kind of features the dictionary is able to catch. The scalar products between the residual and the molecules of the candidate nodes are computed locally, around the position of the molecule, in a search window of size σ. The traversal is over when the algorithm reaches a leaf node. The information about the position and the node of the tree uniquely identifies an atom from the dictionary $\mathcal{D}$. The algorithm goes on until a stopping criterion is met. It could be a predetermined number of atoms, or a threshold on the residual energy.

The choice of the size of the search window σ should depend on the *shape* of the atoms. As a molecule represents its associated subtree, it is reasonable to think that the best position found for a molecule is near the best positions of its children. The search window should reflect the decay of the correlation of the molecules and translated versions of their children (i.e. the decreasing of the cross-correlation).

2.4.2 Complexity Analysis

The complexity of the proposed algorithm highly depends on the structure of the tree. In order to be able to evaluate the complexity of Tree-Based Pursuit, let us first make some hypothesis about the tree. Assume that the number of children per node is a constant k, except for the root node, which has $|a_0|$ children. A tree generated by the algorithm proposed in Section 2.3.2 fulfills these constraints. Let us also suppose that the tree is balanced, meaning that the length of the longest path differs at most by 1 from the length of the shortest path. It ensures that the maximum length of the paths to the leaves is minimized. Under these assumptions, the length of the longest path is $\lceil 1+\log_k \frac{|\mathcal{D}|}{|a_0|} \rceil$, where $|a_0|$ is the number of nodes under the root node and k is the size of the groups formed during the creation of the tree.

Let us first look at the case where the dictionary is not built using translation of generating functions. At the n^{th} iteration, Matching Pursuit would require to compute all scalar products between the atoms of $\mathcal{D}$ and a residual $R^n f$. For a d dimensional signal, the computation of the scalar product needs d multiplications and $d-1$ additions. Thus, the complexity to find one atom is $\mathcal{O}(|\mathcal{D}|d)$. Tree-Based Pursuit reduces this complexity, since the divide and conquer procedure eliminates many possibilities at each level. At the root node, $|a_0|$ scalar products have to be computed. Each other node only requires the computation of k scalar product to find the best child. Thus, the overall complexity is reduced to $\mathcal{O}(|a_0|d + \lceil 1 + \log_k \frac{|\mathcal{D}|}{|a_0|} \rceil kd)$.

For dictionaries built using translation of generating functions, the algorithm has not only to find the best node but also the corresponding position in the signal. At the root node, a full search is done, which is equivalent to Matching Pursuit using the reduced dictionary made of the molecules of the nodes that are located at the first level of the tree. During the rest of the traversal of the tree, a local search in a window of size σ is performed. Let $\tilde{\sigma}$ be the maximal amount of possible positions to test in a window of size σ.

A commonly used and smart implementation of Matching Pursuit consists in using a Fast Fourier Transform to compute all scalar products with shifted atoms. Such an implementation has a complexity of $\mathcal{O}(|\mathcal{D}|d \log d)$ to find the best atom. Each local search has a complexity of $\mathcal{O}(\tilde{\sigma}d)$. Putting it all together, the complexity of the proposed algorithm for finding the best atom is:

$$\mathcal{O}(|a_0|d \log d + (\lceil \log_k \frac{|\mathcal{D}|}{|a_0|} \rceil)\tilde{\sigma}d). \tag{2.15}$$

For reasonable values of the search window σ, the descent trough the tree is negligible regarding the complexity of the initial step. The complexity of Tree-based Pursuit is less affected by the growth of the dictionary, while the complexity of Matching Pursuit increases linearly. However, it has to be noticed that the approximation rate of the Tree-Based Pursuit algorithm decreases when the number of children of the root becomes smaller relatively to the size of the dictionary, as discussed in the next section.

2.5 Consistency analysis

2.5.1 From redundant to block incoherent dictionaries

Most theoretical results in the field of sparse approximations rely on (quasi) incoherent dictionaries. Only little work has been done on highly redundant dictionaries despite their interesting properties for approximation and compression. Interestingly, endowing the dictionary $\mathcal{D}$ with a tree structure can also be thought of as a way to artificially lower the coherence. During the creation of the tree, our clustering algorithm minimizes the coherence among molecules. Thus, even for highly correlated dictionaries, the theoretical results relying on small coherence most probably remain valid at the granularity level of the molecules. In this section, we build upon this idea and analyze the theoretical approximation performance of the algorithm.

The creation of molecules relies on having sub-dictionaries containing highly correlated atoms. The following definition summarizes the constraints ensuring the favorable cases.

Definition 1. *A sub-dictionary $\mathcal{D}_\Lambda$ is reducible to a molecule m_Λ, which is called* representative *if*

- *it's minimal coherence λ_Λ is strictly positive.*
- $\min_{k\in\Lambda} \mid \langle g_k, m_\Lambda \rangle \mid \geq \lambda_\Lambda$.
- $m_\Lambda \in span\ \{\mathcal{D}_\Lambda\}$.

In section 2.3, we have defined an optimality criterion for a molecule relying on the measure of a mean distance that defines a convex set. This implies that standard optimization tools can be applied to find an optimal molecule. Let us now measure the adequation of a molecule regarding the sub-dictionary it represents by :

$$\sigma_\Lambda = \min_{i\in\Lambda} \mid \langle m_\Lambda, g_i \rangle \mid . \tag{2.16}$$

The definition of a representative molecule therefore implies that the minimal coherence of a molecule regarding its associated sub-dictionary is such that $\sigma_\Lambda \geq \lambda_\Lambda$. In other words, adding the molecule m_Λ to its sub-dictionary $\mathcal{D}_\Lambda$ does not change the minimal coherence. This condition defines a subspace of $span\{\mathcal{D}\}$ where the molecule is allowed to exist.

2.5.2 Covering conditions

Since the search is organized along a tree structure, it has to be ensured that the restructured dictionary is still able to cover the full space of the input signal. Tropp has defined a measure of the covering radius of a dictionary [79], as :

$$cover(\mathcal{D}) = \max_{s\neq 0} \min_{i\in\Gamma} \sqrt{1 - (\frac{\mid \langle g_i, s \rangle \mid}{\|g_i\|_2 \|s\|_2})}. \tag{2.17}$$

The relation between the covering radius and the structural redundancy β of a dictionary given in Eq. (2.5) is straightforward. The covering is minimal when β is maximal :

$$cover(\mathcal{D}) = \sqrt{1-\beta^2}. \tag{2.18}$$

We now set the conditions that are necessary for the clustered dictionary to fully cover the signal space. In particular, it is necessary that the molecules at the first level under the root node, cover the signal space. Note that such a requirement is naturally met at other levels of the tree: by the bottom-up construction, each molecule is indeed representative of the related sub-dictionary. The following lemma states a minimal condition on the molecules to ensure that a signal f, which can be represented using atoms from $\mathcal{D}$, can also be represented using only molecules. More precisely it provides a minimal condition, given the parameter β of $\mathcal{D}$, to ensure that the molecules at the first level of the tree cover the same span as the dictionary itself.

Lemma 1. *If the collection of sub-dictionaries $\{\mathcal{D}_{\Lambda_i},\, i = 1, \ldots,\, K\}$ forms a partition of $\mathcal{D}$ and the associated molecules are representative, then span$\{m_{\Lambda_i,\, i=1,\ldots,\, K}\}$ = span $\mathcal{D}$ if*

$$\sigma_{\Lambda_i} > \beta + 2\sqrt{1-\beta} - 1, \forall i. \tag{2.19}$$

Proof. Since the molecules are by construction in the span of their associated sub-dictionaries, the span of the molecules is within the span of the original dictionary $\mathcal{D}$:

$$span\ \{m_{\Lambda_i},\, i = 1, \ldots,\, K\} \subseteq span\ \mathcal{D}. \tag{2.20}$$

In order to ensure that the span of the molecules covers the span of the dictionary, it remains to show that the orthogonal complement of *span* $\{m_{\Lambda_i},\, i = 1, \ldots,\, K\}$ in *span* $\mathcal{D}$ is actually empty.

Let $f \neq 0$ be a signal lying in the span of the dictionary $\mathcal{D}$. Without loss of generality, let f be a unit norm signal. In addition, let the atom $g_0 \in \mathcal{D}$ carry the best one-term approximation of the signal, i.e., $|\ \langle f, g_0\rangle\ | = \max_{i\in\Gamma} |\ \langle f, g_i\rangle\ |$. Suppose the atom g_0 belongs to the sub-dictionary $\mathcal{D}_{\Lambda_0}$ which is represented by the molecule m_{Λ_0}. The distance between f and m_{Λ_0} can be bounded by :

$$\|f - m_{\Lambda_0}\|_2 \leq \|g_0 - m_{\Lambda_0}\|_2 + \|f - g_0\|_2. \tag{2.21}$$

Without loss of generality, assume that $\langle f, g_0\rangle > 0$ and $\langle m_{\Lambda_0}, g_0\rangle > 0$, by construction of the clustered dictionary. Recall that the direction of an atom does not have any impact in terms of approximation rate, so that we can assume positive correlation values. Since all vectors have unit norm, it is possible to rewrite eq. (2.21) as :

$$\sqrt{1 - \langle f, m_{\Lambda_0}\rangle} \leq \sqrt{1 - |\ \langle g_0, m_{\Lambda_0}\rangle\ |} + \sqrt{1 - |\ \langle f, g_0\rangle\ |}. \tag{2.22}$$

We can also bound the last scalar product by :

$$|\ \langle f, g_0\rangle\ | \geq \beta. \tag{2.23}$$

Using equations (2.16) and (2.23), we obtain :

$$\sqrt{1-\langle f, m_{\Lambda_0}\rangle} \leq \sqrt{1-\sigma_{\Lambda_0}} + \sqrt{1-\beta}. \tag{2.24}$$

We would like to show that the projection of the signal f onto the molecule that is representative of the sub-dictionary $\mathcal{D}_{\Lambda_0}$ is never null. In other words, we would like to ensure that, if the best one-term approximation of f lies within $\mathcal{D}_{\Lambda_0}$, then the signal f is never orthogonal to the molecule m_{Λ_0}. Imposing that m_{Λ_0} is not orthogonal to f is equivalent to require that $\sqrt{1-\langle f, m_{\Lambda_0}\rangle} \neq 1$. Using eq. (2.24), this holds whenever

$$\sqrt{1-\sigma_{\Lambda_0}} + \sqrt{1-\beta} < 1, \tag{2.25}$$

which leads to :

$$\sigma_{\Lambda_0} > \beta + 2\sqrt{1-\beta} - 1. \tag{2.26}$$

If this condition is verified, it ensures that $\langle f, m_{\Lambda_0}\rangle > 0$ whenever the signal $f \in \mathcal{D}$ has a component along $g_0 \in \mathcal{D}_{\Lambda_0}$.

If the condition given in eq. (2.26) is true for all sub-dictionaries of the first level of the tree (that form a partition of $\mathcal{D}$), then $\nexists f \in \mathit{span}\ \mathcal{D}$ such that $\langle f, m_{\Lambda_i}\rangle = 0$, $\forall i$. Hence $\mathit{span}\ \mathcal{D} = \mathit{span}\ \{m_{\Lambda_i},\ i = 1, \ldots,\ K\}$. □

When the formula of Lemma 1 holds, we can treat the set of molecules as a genuine dictionary. Let $\{\mathcal{D}_{\Lambda_i}\}$ form a partition of $\mathcal{D}$ and let $\mathcal{D}_M = \{m_{\Lambda_i}\}$ be the dictionary made of the molecules. This dictionary has an associated characteristic parameter β_M. For any signal $f \in \mathit{span}\ \mathcal{D}$, we thus can lower bound the projection on the molecules :

$$\max_{m_i \in \mathcal{D}_M} |\langle f, m_i\rangle| \geq \beta_M ||f||. \tag{2.27}$$

This also leads to :

$$\max_{m_i \in \mathcal{D}_M} |\langle f, m_i\rangle| \geq \beta_M \max_{i \in \Gamma} |\langle f, g_i\rangle|. \tag{2.28}$$

Obviously $\beta_M \leq \beta$. It would also be interesting to characterize the (cumulative) coherence of the dictionary. In the next section we show that Tree Based Pursuit benefits from representative molecules and is able to identify the signal at the granularity level of its representative sub-dictionaries.

2.5.3 Recovery Condition

In the previous section, we have set the conditions for the tree structured dictionary to cover the span of the original dictionary $\mathcal{D}$. We now derive a condition for the search algorithm to choose consistent molecules given a signal f, that is a linear combination of vectors in $\mathcal{D}$. Let the signal f have an exact representation using atoms from the dictionary $\mathcal{D}$:

$$f = \sum_{i \in \Omega} c_i g_i, \tag{2.29}$$

where Ω is a subset of indices.

Tropp [78] derived a minimal condition that guarantees that Orthogonal Matching Pursuit and Basis Pursuit recover Ω, where Ω is the smallest set such that eq. (2.29) holds. We now show that this recovery condition holds true for TBP at the level of representative molecules of a very redundant dictionary. Let Φ be a matrix whose columns contain the atoms that are in Ω. The signal can be written as $f = \Phi A$, where the vector A contains the weights c_i relative to atoms in Ω.

Let f_k be the approximation of f after k iterations of Tree-Based Pursuit. We write $f_k = \Psi_k A_k$, where Ψ_k contains the atoms found by Tree-Based Pursuit and A_k the corresponding weights. Since we do not impose any restriction on the cumulative coherence of the dictionary, we cannot directly apply the results developed in [78], which typically use the cumulative coherence for an estimation of the exact recovery condition. We do not necessarily intend to recover exactly the atoms in Φ, but we rather want to ensure that the atoms found by Tree-Based Pursuit are close to the optimal ones (and in particular, in the same sub-dictionaries). We focus on the decision taken by Tree-Based Pursuit at the root of the tree and want to guarantee that it never chooses a node that does not contain at least one atom from Ω in its subtree.

If after k iterations of Tree-Based Pursuit, the decisions at the root node are always *correct*, no atom from Ψ_k is located in a subtree that does not contain an atom from Ω. Let Φ_k be a matrix containing the distinct atoms from Φ and Ψ_k. Similarly, the index set Ω_k is the set of atoms present in Φ_k. As it has been discussed, due to the bottom-up construction of the tree, the critical step consists in choosing the correct molecules at the first level of the tree. Assume once again that the sub-dictionaries $\{\mathcal{D}_{\Lambda_i}\}$ form a partition of the dictionary, and that each sub-dictionary is reduced to a molecule m_{Λ_i}. We say that m_{Λ_i} is a *good* molecule if it represents at least one atom participating in f. The matrix M_G contains all *good* molecules in its columns. Similarly, M_B contains the *bad* molecules of the first tree level in its columns. The following theorem states the necessary conditions for the tree-based pursuit algorithm to choose the correct molecule at the first level of the tree.

Theorem 1. *If the hypothesis of Lemma 1 holds true, then Tree-Based Pursuit chooses a good molecule at the first level of the tree, at iteration k, if*

$$\max_{m \in M_B} ||\Phi_k^+ m||_1 < \beta_M, \tag{2.30}$$

where Φ_k^+ is the Moore-Penrose pseudo-inverse of Φ_k.

Proof. The proof of Theorem 1 is an extension of Tropp's Recovery Condition [78] and we provide here only the differences to the proof given in [78]. Assume that at each iteration $i < k$, Tree-Based Pursuit has chosen a good molecule at the first level of the tree. It has to be noted that the atoms in Ω all belong to subtrees of nodes associated to *good* molecules. Under the assumption that we have chosen only *good* molecules, the atoms in Ω_k also belong to subtrees of nodes associated to *good* molecules. The residual signal $r_k = f - f_k$ can be exactly represented as $r_k = \Phi_k A_k^R$, where A_k^R contains appropriate weights. The vectors $M_B^* r_k$ and $M_G^* r_k$ list all possible scalar products of the residual r_k with, respectively, the *bad*

and *good* molecules (M^* stands for the adjoint of M). The aim is to find a condition that ensures that the current step also recovers a good molecule. A *good* molecule is therefore chosen by the search algorithm if :

$$\frac{||M_B^* r_k||_\infty}{||M_G^* r_k||_\infty} < 1. \tag{2.31}$$

If the hypothesis of Lemma 1 holds true, then the lower bound on the projection on the molecules given in Eq. (2.28) can be used to further develop the left-hand side of the previous equation. We can write :

$$\frac{||M_B^* r_k||_\infty}{||M_G^* r_k||_\infty} \leq \frac{||M_B^* r_k||_\infty}{\beta_M ||\Phi_k^* r_k||_\infty}. \tag{2.32}$$

The last steps of the proof are analog to Tropp's recovery condition [78] proof, which finally leads to the conservative condition :

$$\frac{1}{\beta_M} \max_{m \in M_B} ||\Phi_k^+ m||_1 < 1. \tag{2.33}$$

□

One could further apply Tropp's estimate of eq. (2.33) in terms of the cumulative coherence [78] of the set of molecules, in order to obtain a condition that would depend on the set of molecules only (and not on the unknown optimal set M_B). This estimate requires the set of molecules to be quasi-incoherent. Note that this is very likely to be the case here, but it would even be better to actually prove how μ_1 behaves as we climb up the granularity level of the tree. Finally, note that the recovery condition itself holds at a coarser level than in previous works : Tree-Based Pursuit recovers only molecules that are involved and not the individual atoms. On the other hand, this allows to shift the incoherence constraint to the molecules and work with a possibly highly correlated dictionary.

2.6 Experimental Results

2.6.1 1-D signals

This section now illustrates the Tree-Based Pursuit algorithm, and compares its performances to Matching Pursuit. We present results for both 1-D and bi-dimensional signals (i.e., images). Often, in practice, the dictionaries are built using shift invariant generating functions that can be translated at any place in the support of the signal. In order to illustrate the generality of the method, let us first consider a dictionary $\mathcal{D}$ made of random atoms. It represents the worst case for the construction of the tree. The size of the dictionary is $|\mathcal{D}| = 4000$. The atoms are real vectors of size $d = 40$. Thus, the dictionary is 100 times redundant. Each sample has uniform probability between 0 and 1 and the atoms are normalized to have unit energy. The complexity of Tree-Based Pursuit mostly depends

on the number of molecules under the root node. Thus, 20 trees have been generated for different numbers of nodes at the first level of the tree, $|a_0|$ ranges from 200 up 4000 by steps of 200. Two sets of trees have been generated depending on the decision rule to create the molecules (*weak* or *strong*).

In order to evaluate the performances of Tree-Based Pursuit, 100 test signals have been randomly generated using the same procedure as for the dictionary. Each individual signal has been approximated using $N = 30$ atoms from the dictionary $\mathcal{D}$. Figure 2.3 illustrates the mean approximation error achieved. It has to be noticed that for $\frac{|a_0|}{|\mathcal{D}|} = 1$, Tree-Based Pursuit is equivalent to Matching Pursuit. The ratio $\frac{|a_0|}{|\mathcal{D}|}$ reflects roughly the complexity of Tree-Based Pursuit regarding Matching Pursuit. The error decreases as the number of nodes at the first level of the tree increases.

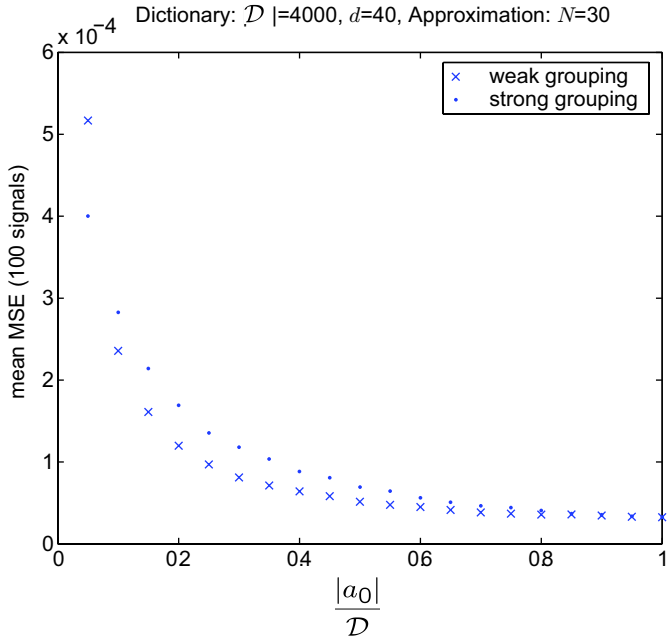


Figure 2.3 – *Mean approximation error of random signals using a random dictionary.*

Random dictionaries are seldom used to approximate signals. Consider now a dictionary made of real Gabor functions, as in [51] :

$$g_{u,s,\xi,\phi}(t) = C_{u,s,\xi,\phi} g(\frac{t-u}{s}) \cos(2\pi\xi(t-u) + \phi), \tag{2.34}$$

with

$$g(t) = \frac{1}{\sqrt{s}} e^{-\pi t^2}. \tag{2.35}$$

The normalizing constant $C_{u,s,\xi,\phi}$ is such that the corresponding atom is of unit energy. The parameter u is the position, s is the scale, ξ represents the frequency and ϕ is the phase. Figure 2.4 presents 3 atoms of such a dictionary, and the representative molecule, which is the eigenvector associated to the biggest eigenvalue of $A_\Lambda A_\Lambda^*$, as discussed in Section 2.3. Figure 2.5 shows the time-frequency representations of the atoms and the molecule of

Figure 2.4. We can observe that the molecule indeed provides global information about all the atoms, and nicely summarizes the characteristics of the sub-dictionary.

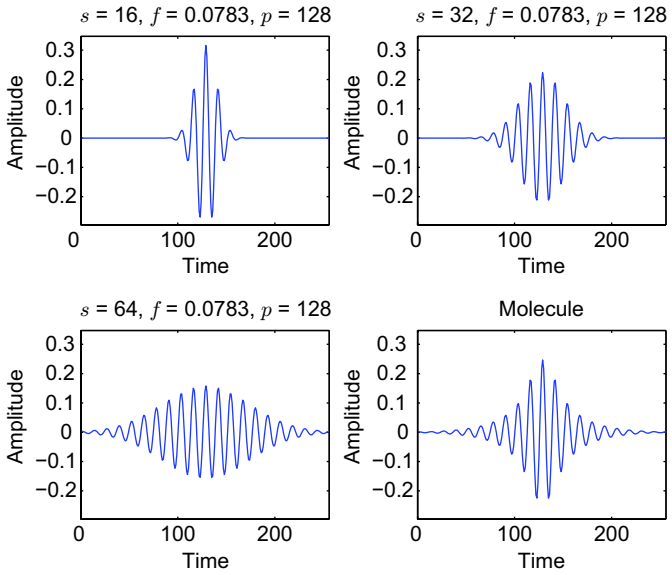


Figure 2.4 – *Representing a group of atoms by a molecule. From top-left to bottom-left: Real Gabor atoms with same frequency f and position p but with different scales s. Bottom-right: the molecule is the eigenvector associated to the biggest eigenvalue of $A_\Lambda A_\Lambda^*$.*

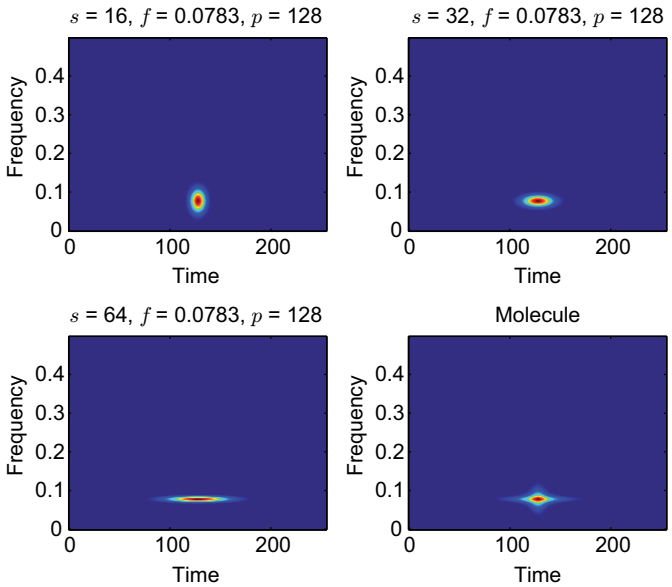


Figure 2.5 – *Time-Frequency plane of the atoms and the molecule presented in Figure 2.4.*

In our experiments, we used a dictionary built on real Gabor atoms with size 256, where the phase ϕ is set to zero in eq. (2.34). We used 200 different frequencies uniformly spread over the interval of normalized frequencies [0 0.5] and the scales are dyadic. The overall

size of the dictionary is 1600 (i.e. 1600 times redundant) , without taking into account all possible shifts, which are not considered during construction of the trees. The translation parameters are however computed by the search algorithm. Figure 2.6 shows a part of an example tree built on the multiscale Gabor dictionary, where we only use *centered* versions of the atoms.

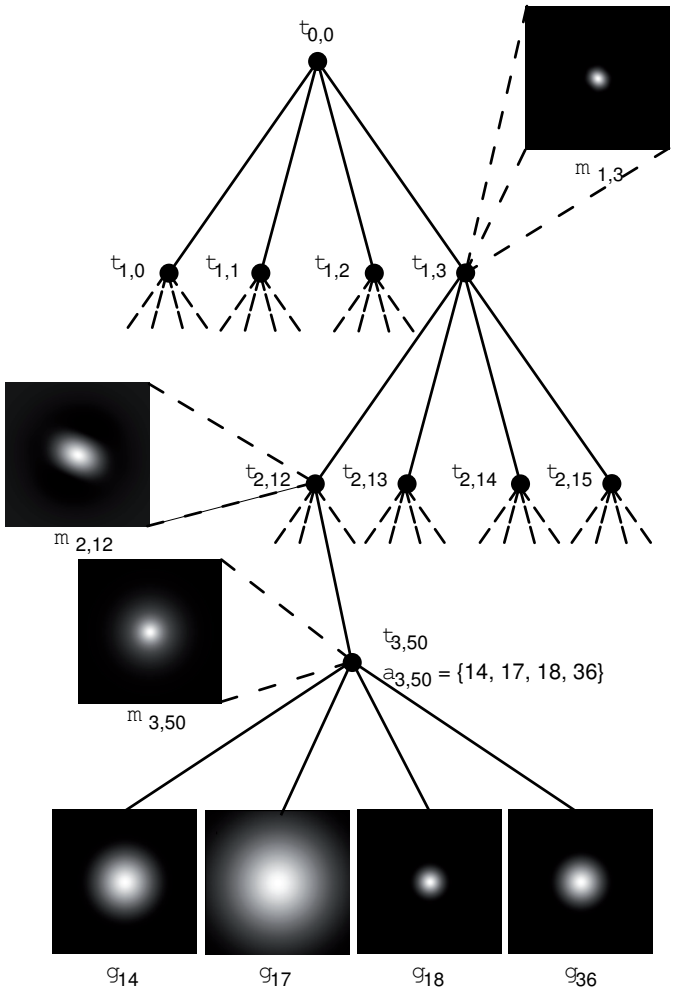


Figure 2.6 – *Tree structure on top of a multiscale Gabor dictionary.*

We now compare the performance of the Tree-based Pursuit algorithm, for different tree constructions, with Matching Pursuit. The reference Matching Pursuit computes all possible convolutions in the frequency domain by using a Fast Fourier Transform. Tree-Based Pursuit uses the same Matching Pursuit implementation at the initial step for the first level of the tree. This technical choice makes it possible to compare the complexity of both algorithms.

Numerous tree structured dictionaries have been generated for different values of the distance threshold δ, using the grouping strategy given in Algorithm 2.3.3, with a *weak* decision rule for clustering of the atoms. We selected three different trees, with δ values

[0.36 0.75 0.99]. This corresponds respectively to minimal values of [0.8 0.5 0.1] of the scalar product between two molecules to form a cluster. The value of δ determines $|a_0|$, the number of nodes at the first level of the tree. In these particular cases, the trees respectively present 240, 51 and 11 nodes at the first level under the root node. Moreover, under the assumption that the trees are balanced, their average depth would be 3, 5 and 8 in the order of increasing values for δ. Figure 2.7 presents the molecules at the first level of the tree created with $\delta = 0.99$, while Figure 2.8 illustrates the corresponding time-frequency diagrams.

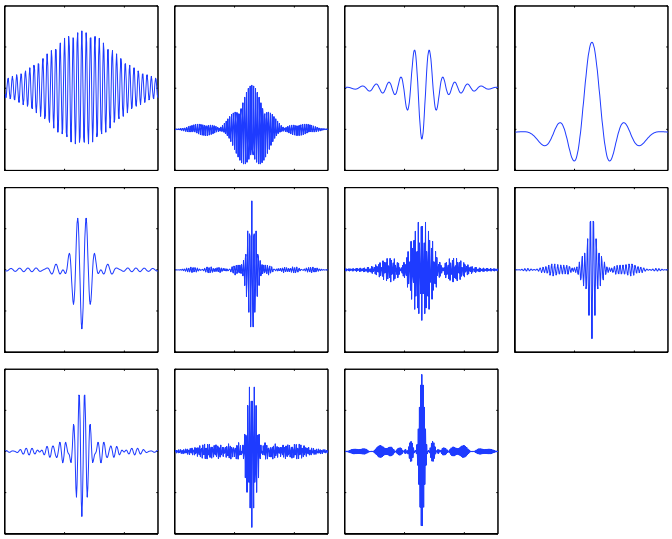


Figure 2.7 – *Molecules associated to the nodes located at the first level of the tree created with a grouping threshold* $\delta = 0.99$.

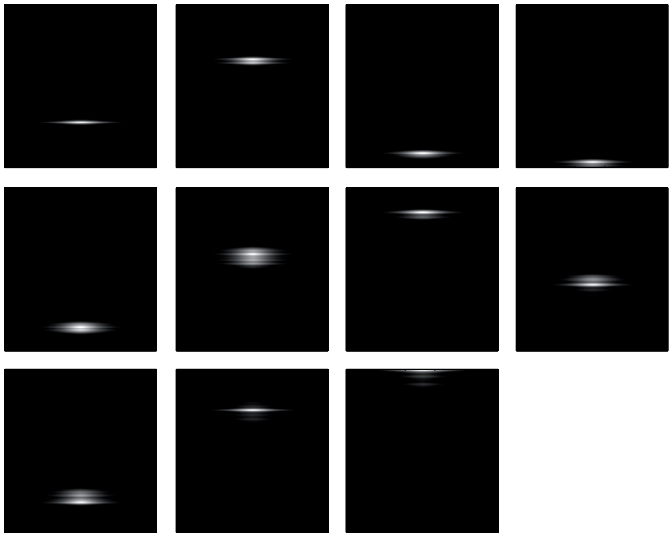
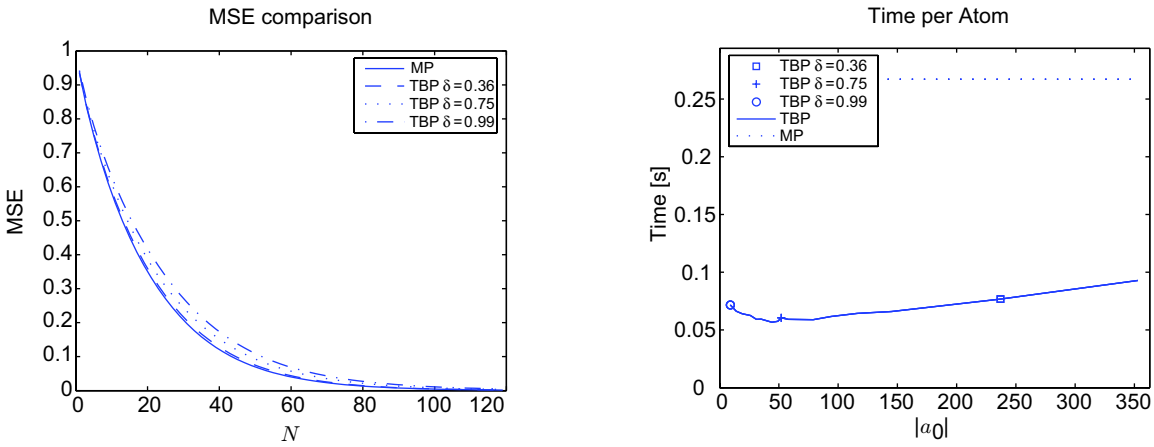


Figure 2.8 – *Time-frequency distributions corresponding to the molecules exhibited by Figure 2.7.*

We can now compare the approximation performance, and the computational complexity of Tree-based Pursuit, as opposed to Matching Pursuit. The size of the search window σ is 5. Part (a) of Figure 2.9 compares the mean square error obtained with Matching and Tree-Based Pursuit using the different trees defined above. The results have been averaged over 100 zero-mean random signals with Gaussian distribution of unit variance. When using trees created with small values for δ, the results are very close to the reference Matching Pursuit. Recall that small values of δ impose very strict constraints on the clustering of atoms and molecules, which may result in a large number of molecules at the first level of the tree. The computation time depends on δ, since it depends on the amount of nodes at the first level of the tree $|a_0|$, as discussed in Section 2.4. Experimental results confirm the complexity analysis in part (b) of Figure 2.9. Indeed, if we compute a linear approximation of the Tree-based Pursuit computation time as a function of $|a_0|$, in a mean square sense, it intersects the Matching Pursuit computation time around $|a_0| = 1618$ (the dictionary contains 1600 atoms). This shows that most of the complexity of Tree-based Pursuit lies in the full search at the first level of the tree; after this initialization step, the cost of the traversal of the tree can be considered as negligible regarding the initial search.



(a) (b)

Figure 2.9 – *(a) Comparison of the error produced by the proposed algorithm when using different bounds for the grouping. (b) Comparison of the complexity between Matching Pursuit and Tree-Based Pursuit, when using different values for δ during the creation of the tree structure.*

2.6.2 Extension to multi-dimensional signals

This section extends the analysis of the Tree-based Pursuit algorithm to images. Reduction of the complexity in the case of multidimensional signal is even more crucial than for 1-D signals. We use a dictionary that is built on gaussian generating functions that are scaled, rotated and translated [30]. The first generating function is a Gaussian as given by eq. (2.36), which suits well the task of capturing the low-frequency parts of natural images. The second generating function, given in eq. (2.37) is made of a Gaussian in one direction and its second derivative in the other direction. It has a good ability to capture edges in

images and is spatially and frequencially well located.

$$g_1(x,y) = \frac{1}{\sqrt{\pi}} \exp -(x^2+y^2). \tag{2.36}$$

$$g_2(x,y) = \frac{2}{\sqrt{3\pi}}(4x^2-2) \exp -(x^2+y^2). \tag{2.37}$$

In our experiments, the atoms using g_2 as generating function have translation parameters that take any positive integer value smaller than the size of the image. The rotation parameter varies by increments of $\frac{\pi}{18}$. The scaling parameters are uniformly distributed on a logarithmic scale from one up to an eighth of the size of the image, with a resolution of one third of octave. The scaling along the second derivative part is always smaller. For the pure Gaussian atoms, the translation parameters can take the same values, the scaling is isotropic and varies from $\frac{1}{32}$ to $\frac{1}{4}$ of the size of the image on a logarithmic scale with a resolution of one third of octave. Due to isotropy, rotation is obviously useless for this kind of atoms.

The dictionary is made of 514 generating functions that can be placed anywhere on the support of the signal (i.e. the dictionary is 514 times redundant). Algorithm 2.3.3 was used with a *strong* decision rule to create the molecules; 20 different trees have been created with different values of δ, chosen to create trees having values for $|a_0|$ ranging from $0.05|\mathcal{D}|$ up to $\mathcal{D}$ by steps of $0.05|\mathcal{D}|$.

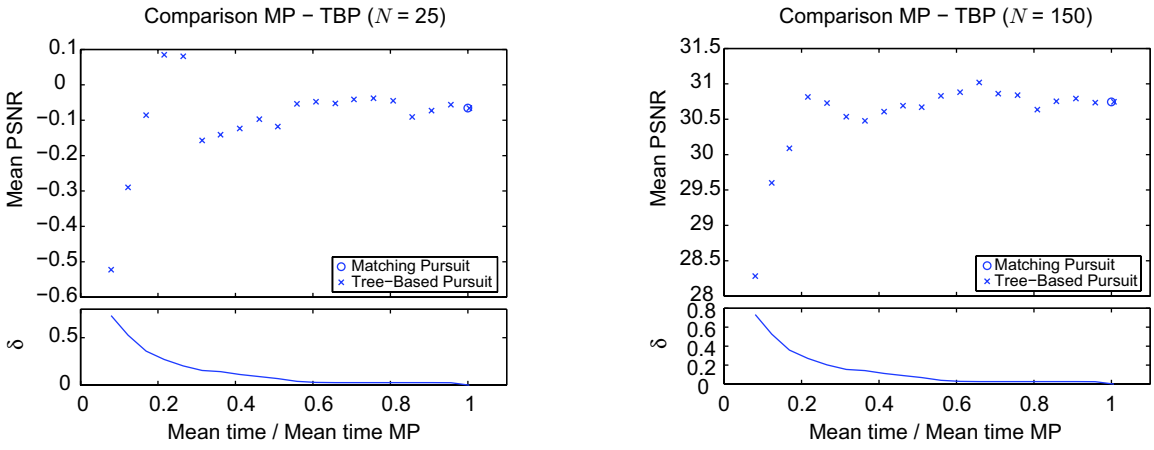
Figure 2.10 presents a comparison of the approximation error of Tree-based Pursuit and Matching Pursuit averaged over 120 natural images of size 16×16. The mean time to find one atom has also been computed. As explained in section 2.4, the computational time mostly depends on the number of nodes at the first level of the tree. Figure 2.10 shows that for a computational time of about 20 percent of the reference Matching Pursuit, the errors are comparable. When using only a few atoms, it also happens that Tree-Based Pursuit performs better. The used dictionary represents a favorable case for the creation of a tree structure as the atoms corresponding to edges in the same direction can be efficiently represented by a unique element without loosing the edge detection ability. This dictionary has a flavor of the ideal case represented by block incoherent dictionaries.

The lower parts of the results presented in Figure 2.10 present the value of δ used for the creation of the corresponding trees. Tree-Based Pursuit is equivalent to Matching Pursuit when using trees created with a value of $\delta = 0$. However, due to a more complex data structure to handle, the computational time is slightly higher.

2.7 Application: Very low bit rate face coder

2.7.1 Motivations

Tree-Based Pursuit provides good approximations at much lower costs than Matching Pursuit. In this section we tackle the problem of coding human faces at very low bit rate. Coding schemes using redundant dictionaries are very efficient to achieve low bit rates



(a) (b)

Figure 2.10 – *Performance of Tree-Based Pursuit compared to Matching Pursuit for different trees. The x axis corresponds to the mean time per atom divided by the mean time per atom for Matching Pursuit. The lower part of the figures presents the value of δ used during the creation of the tree. Part (a), mean error achieved using 25 atoms. Part (a), mean error achieved using 150 atoms.*

whilst having limited visual artifacts [29] at the cost of finding good approximations. In addition, very low bit rate coding of human faces is an application that would normally take place in a context where a huge number of faces has to be treated. Thus, the encoding time has to be low. It represents a typical case where Tree-Based Pursuit enables to benefit from sparse representations while keeping an acceptable computational complexity.

Compression matters for security applications as it may be of great interest to be able to save the compressed images on cheap memory cards. It permits to integrate the digital face into different supports as ID cards or passports. It could also be of great interest to be able to store many faces on portable devices. Our coding scheme achieves the desired compression ratio by mean of sparse approximation using an adapted redundant dictionary.

2.7.2 Coding scheme

In [63], the authors propose to use a Laplacian Pyramid to split the image into a *coarse* and a *detail* version. The first part is just quantized and the *detail* version is approximated using Matching Pursuit and an adapted dictionary. Most of the important visual information is located in the *detail* version.

The proposed coding scheme has been inspired by [63]. Whilst their coding scheme works for any kind of image, we wanted to take advantage of the knowledge that we have to encode faces. Our assumptions are that the size of the images is fixed and that the position and the orientation of the faces are always the same.

Figure 2.11 presents the coding scheme. First, the image I passes trough a filtering stage. Let I_L be the low pass version and I_H the high pass one such that $I = I_L + I_H$. The two different parts will be approximated and encoded independently. Given a target

bit rate, a Rate-Distortion optimization is used to compute the optimal number of bits to allocate for the different versions. A Rate-Distortion based method is also applied to obtain the optimal parameters for the quantization steps. Finally, the approximation is encoded in a bitstream.

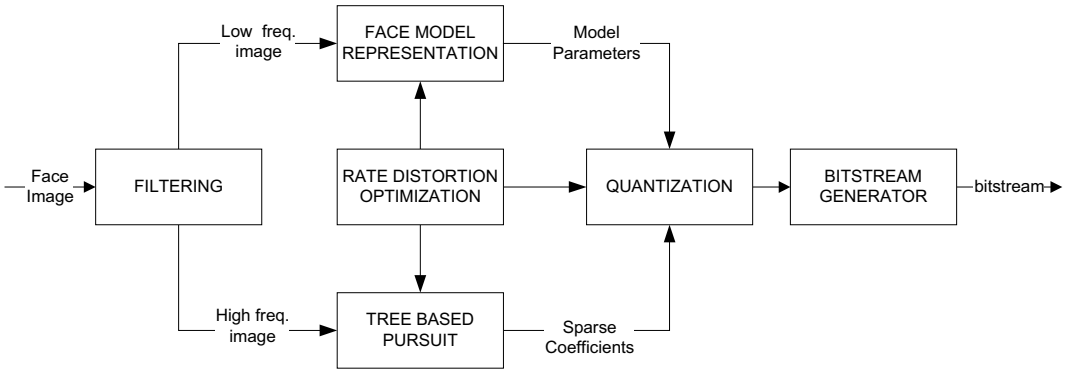


Figure 2.11 – *Coding scheme of the proposed low bit rate face coder.*

Low frequencies coding

All faces are located roughly in the same position inside the images. Thus, the versions containing the low frequencies are looking very similar and are far from being random points in the image space. We take advantage of this by learning an adapted orthogonal base. We used the Principal Component Analysis to get the vectors that match the distribution of the low frequency version. In the case of face images, the eigenvectors are called Eigenfaces [81] and noted $\mathcal{E} = \{e_i\}$. This technique has been used to detect or classify faces. We learned the Eigenfaces from a set of face images. The ones associated with the biggest eigenvalues are presented in Figure 2.12.

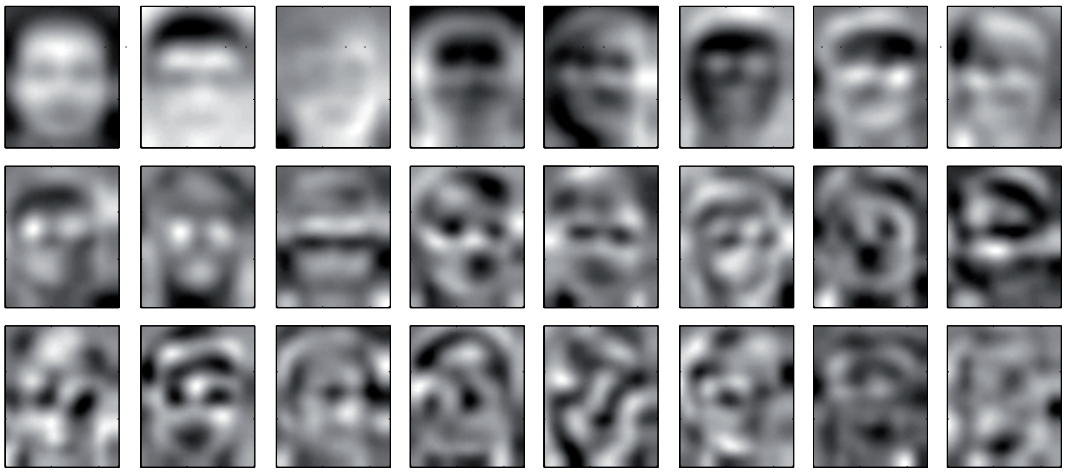


Figure 2.12 – *First Eigenfaces used to encode the low frequencies in the face images.*

The low pass version of the image is approximated as a weighted sum of N_e learnt Eigenfaces. Given the total number of bits, a Rate Distortion optimization allocates a certain bit budget b_L for coding the low frequencies. The best approximation with N_e terms is found by taking the Eigenfaces associated with the projections having the biggest amplitude i.e.

$$I_L \approx \sum_{i=1}^{N_e} c_i e_i, \tag{2.38}$$

where the projections a_i are sorted in decreasing order of magnitude.

In order to be able to encode the amplitudes, they have to be quantized. Let $Q_{\Delta_e}()$ be a uniform quantizer with quantization step Δ_e. The approximated low frequency image $\hat{I}_L$ is :

$$\hat{I}_L \approx \sum_{i=1}^{N_e} Q_{\Delta_e}(c_i) e_i, \tag{2.39}$$

where Δ_e and N_e are such that $\|I_L - \hat{I}_L\|_2$ is minimum given the bit budget b_L.

High frequencies coding

The approximation of the low frequencies is fast as it is done with an orthogonal base that is learnt off-line. We want to achieve very low bit rates and the efforts have to be concentrated on the most *difficult* information to code. Thus, the high frequencies are approximated with a redundant dictionary of geometric meaningful functions.

The dictionary is constructed from generating functions $g(x_1, x_2)$ that are submitted to different geometric transforms. The basis function is made of a Gaussian in one direction and its derivative in the other. The possible geometric transforms are precisely described in [63] :

Translation T_{p_1,p_2} Modify the position of the atom in the image.

$$T_{p_1,p_2} g(x_1, x_2) = g(x_1 - p_1, x_2 - p_2)). \tag{2.40}$$

Rotation R_θ Changes the orientation of the atom.

$$R_\theta g(x_1, x_2) = g(\cos\theta x_1 - \sin\theta x_2, \sin\theta x_1 + \cos\theta x_2). \tag{2.41}$$

Bending B_r Permits to adapt the atom to the contours by bending the x_2 axis with a radius r.

$$B_r g(x_1, x_2) = \begin{cases} g(r - \sqrt{(x_1 - r)^2 + x_2^2}, r \arctan(\frac{x_2}{r - x_1})) & \text{if } x_1 < r. \\ g(r - |x_2|, x_1 - r + r\frac{\pi}{2}) & \text{if } x_1 \geq r. \end{cases} \tag{2.42}$$

Scaling S_{s_1,s_2} The scaling is anisotropic i.e. different parameters are used for both directions.

$$S_{s_1,s_2}g(x_1,x_2) = g(\frac{x_1}{s_1},\frac{x_2}{s_2}). \tag{2.43}$$

An atom is defined by the parameters of the previously described transforms; the atom is normalized to have unit norm.

$$g_{p_1,p_2,\theta,r,s_1,s_2}(x_1,x_2) = \frac{T_{p_1,p_2}R_\theta B_r S_{s_1,s_2}g(x_1,x_2)}{\|T_{p_1,p_2}R_\theta B_r S_{s_1,s_2}g(x_1,x_2)\|_2}. \tag{2.44}$$

Figure 2.13 illustrates the different geometric transforms modifying the shape of the atom. The dictionary is made of all atoms that can be created with these transforms. Even when using small sets of possible values for each parameters, we generate highly populated dictionaries.

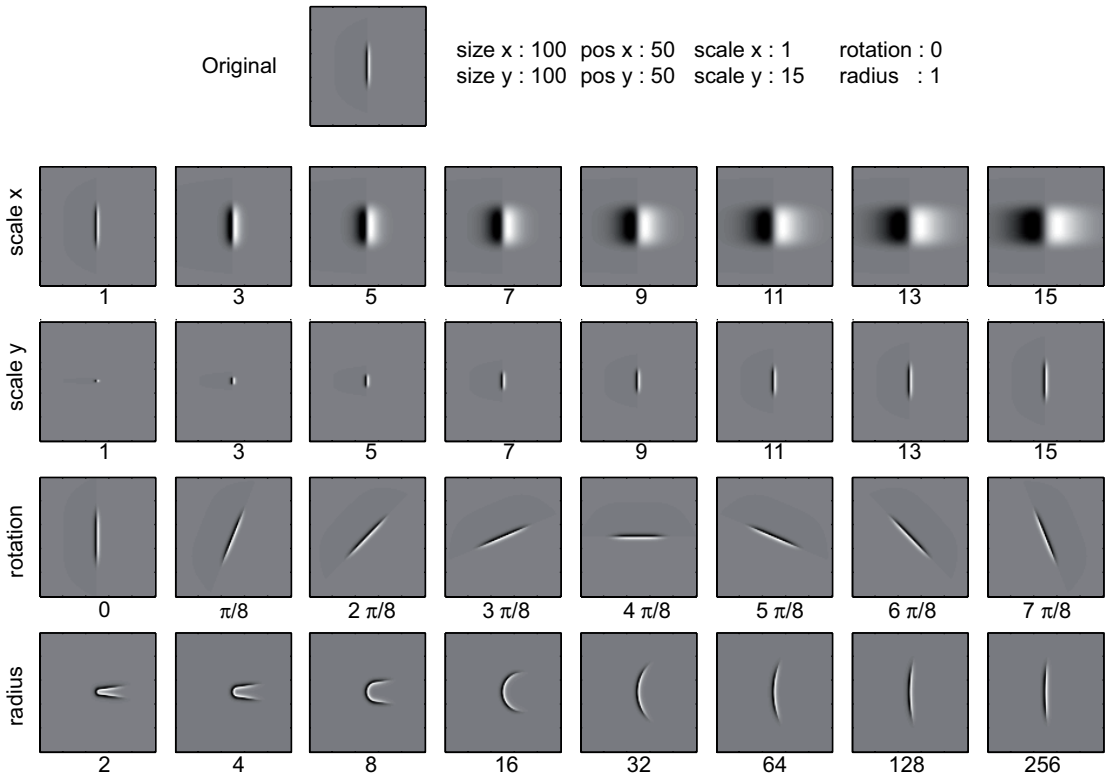


Figure 2.13 – *The atom on top of the figure is submitted to different changes of the parameters of the transforms. The first and the second row present changes of parameters the scaling transform* (2.43). *The third row presents different rotations* (2.41). *The last row presents changes of the radius related to the bending transform* (2.42).

The cardinality of the dictionary has an important incidence on the computational complexity of algorithms like Matching Pursuit. This motivates the use of Tree-Based Pursuit as algorithm to find a sparse approximation of the image containing the high frequencies as it permits to represent similar atoms by a single one. The number of terms

to use is given by a Rate-Distortion optimization and depends on the total number of bits allocated. The found parameters of the atomic decomposition are encoded into the stream.

2.7.3 Results and discussion

Figure 2.14 compares the proposed scheme with a state of the art JPEG2000 coder [70]. The first lines only show results of our coder as the reference coder was not able to achieve these compression ratios. Even with a very limited number of atoms to represent the high pass version, a human user is able to recognize the person. Additionally, using geometric functions as atoms do not introduce visually disturbing artefact.

Security is a major issue in our modern society. Specifically, identity authentication is one of the most important aspect of security management. Biometric identification technology represents an extraordinary automatic mean to positively identify a person. Due to its absence of contact and non-invasiveness, face recognition is viewed as an excellent solution for biometric authentication for widely spread applications such as authentication for banking, security system access, advanced video surveillance, video annotation.

The size of biometric information databases imposes drastic compression requirements on storage and transmission of identification data. Both the authentication and the compression process have to reduce the dimensionality of the data to keep only the fundamental information regarding their aim. The previous results showed the ability of sparse representation to achieve high compression ratios. In [52], we proposed to use the parameters of a sparse representation to perform face authentication using a Multi-Layer Perceptron. The obtained results were compared with PCA and LDA systems on the multi-modal benchmark database BANCA. LDA is performing slightly better but we have shown that using the previously presented dictionary permits to obtain sparse representations whose parameters may be used for face authentication whilst providing outstanding performance in compression. Additionally, in [40], we explored the use of sparse representations for different security applications and concluded that using simple geometric manipulations on the atoms lead to interesting results.

2.8 Discussion

This chapter presented a generic algorithm to reduce the computational complexity of pursuit algorithms. Hierarchical clustering of dictionary atoms in molecules has been proposed, as an efficient structuring of large set of functions. The molecules represent a sub-dictionary of highly correlated atoms and are used to create a tree structure from an arbitrary highly redundant dictionary. A tree-based pursuit algorithm is then proposed, which exploits the tree structure, resulting in a computational complexity that is significantly lower than the classic pure greedy algorithm. We experimentally showed that the reduction in complexity does not imply a large penalty in approximation rate. It is shown also that Tree-Based Pursuit recovers coarse structures of the signal, even for highly redundant dictionaries, thanks to the hierarchical clustering into sufficiently incoherent dictionaries of molecules. Finally,

practical applications are often based on highly redundant dictionaries, whose properties are however poorly studied. On the other hand, the class of *incoherent* dictionaries has been widely studied, but is rarely used in practical applications. Our study tries to bridge that gap, by demonstrating that, from a molecular point of view, it is possible to apply the approximation results for *incoherent* dictionaries to highly redundant dictionaries.

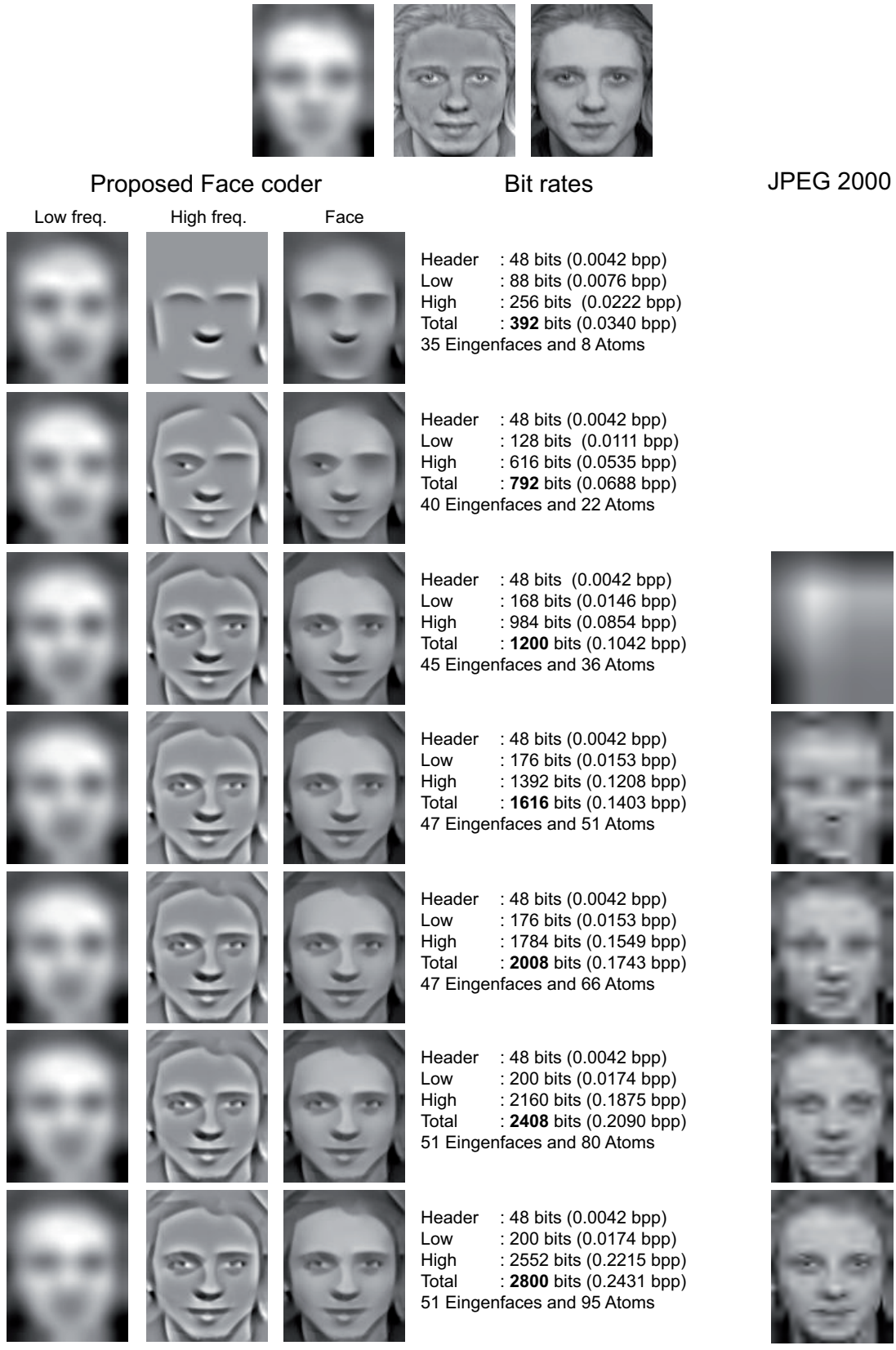


Figure 2.14 – *A face is encoded and decoded using the proposed scheme. The images containing the lower and the higher frequencies are scaled for this presentation. A JPEG 2000 encoded image having the same bit rate is presented whenever it was possible.*

3 Dictionary Learning

3.1 Motivations

We previously stated that expansions using redundant dictionaries of functions is a powerful tool for approximation; few terms capture most of the information of a signal. Typical tasks in signal processing as analysis, dimensionality reduction, de-noising or compression may take advantage of the resulting sparsity.

The properties of the signal, dictionary and algorithm, are tightly linked. Often, natural signals have highly complex underlying structures which makes it difficult to explicitly define the link between a class of signals and a dictionary. This chapter presents a learning algorithm that tries to capture the underlying structures. In our approach, instead of considering atoms having the same support as the signal, we propose to learn small generating functions, each of them defining a set of atoms corresponding to all its translations. This is notably motivated by the fact that natural signals often exhibit statistical properties invariant to translation, and that using generating functions allows to generate huge dictionaries while using only few parameters. In addition, fast convolution algorithms can be used to compute the scalar products when using pursuit algorithms.

The ability of a dictionary to approximate signals is often measured using extreme properties of the dictionary as the coherence μ which is used to provide different bounds on the energy of the residual. The minimal condition needed for a dictionary to be able to represent any signal is to span the entire signal space. Intuitively, the more a dictionary is populated, the better it is for finding efficient short terms representations. The adequation of a dictionary to a class of signals may also be measured by its ability to lead to sparse representations. Sparsity is a very ambiguous concept and a lot spareness measures exist; they may lead to very different results [41]. Thus, measuring the performance of a dictionary

may also be subject to ambiguities. In this chapter, the presented algorithm does not try to maximize directly the adequation of the dictionary but seeks waveforms representing features present in the learning signals.

In section 3.2, we review existing methods to learn overcomplete dictionaries. In section 3.3, we formalize the problem of learning generating functions, and propose an iterative algorithm to learn successively adapted atoms. The proposed algorithm is called MoTIF which stands for *Matching of Time-Invariant Filters*. This work was done in collaboration with Sylvain Lesage and Rémi Gribonval. In order to illustrate the algorithm, this section provides a comprehensive example of learning an overcomplete dictionary for natural images. In section 3.4.1, we present an experiment on the recovery of underlying atoms. In section 3.4.2, we show with another experiment the ability of this learning method to give an efficient dictionary for sparse approximations. Different researchers started from this algorithm and proposed new solutions to increase its efficiency or to use it in different situations. Section 3.5 presents these solutions as well as an application using MoTIF in the field multi-modal signal processing. We conclude in section 3.6 on the benefits of this new approach.

3.2 Introduction

In the previous chapter, we stated that finding a good approximation is a difficult task. In this chapter, we now state that finding a good dictionary is also a difficult task. Generally, to get around this problem, people often choose collections of *allround* atoms as Gabor functions for example. Instead of taking a unique well known collection of functions, it is also possible make combinations of N different dictionaries as follows :

$$\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_N. \tag{3.1}$$

For example, in [75], a natural image is considered as being the sum of a piecewise smooth images and textures. The two different parts have their own specific dictionary for which the corresponding classes of signals admit very sparse decompositions. A Curvelet Transform [74] is used for the smooth part and the textures are represented using the Discrete Cosine Transform. Additionally, a method, named Morphological Component Analysis (MCA), has been designed to separate both parts knowing that they have a sparse representation in their corresponding dictionaries.

Unlike the previous example, if often happens that comprehensive *a priori* information about the class of signals to decompose is not at disposal. Taking an *allround* redundant dictionary is still possible but it is not likely to be the best solution. Let $\mathcal{D}$ be a dictionary and D a matrix whose columns contain the atoms.

Definition 2. *A dictionary $\mathcal{D}$ is good for a class of signals $\mathcal{C}$ if*

$$\forall f \in \mathcal{C} \ , \ \exists c \ s.t. \ Dc = f + \epsilon, \tag{3.2}$$

where c is sparse and ϵ is small.

The concept of sparsity as well as the notion of acceptable error are very subjective and context-dependant. There is not a unique definition of a good dictionary. This simple fact would already lead to a huge variety of solutions for finding *good* dictionaries. Additionally, when using learning algorithms, there are practical constraints intrinsically linked to the problem itself :

- **Limited computational abilities.** In particular it is not possible to solve P_0 in polynomial time.
- **Limited memory.** Practical solutions do not exist to manage too voluminous amounts of data, which is a limiting factor for learning algorithms.
- **The class of signals $\mathcal{C}$ is not fully accessible.** In this chapter, we suppose to have access to a set $\mathcal{F} = \{f_n\}_{n=1}^N$ of N training signals of size S_f.

Generally, overcomplete dictionary learning algorithms consider that each training signal f_i admits a approximation as follow :

$$f_i = Dc_i + \epsilon, \tag{3.3}$$

where the vector c_i is sparse and ϵ is additive Gaussian white noise with variance σ. The different definitions of sparsity leaded to diverse dictionary learning algorithms. They can roughly be divided into two different classes.

- **Probabilistic sparsity model.** The coefficients c_i are supposed to be generated by a Random Variable that has a sparse Probability Density Function. [46, 48, 49, 59–61]
- **Algorithmic dependant sparsity models.** The changes in the dictionary made by the algorithm are function of coefficients c_i that are the outcomes of an approximation algorithm. [1–4, 26, 27, 43]

Surprisingly, researcher working on sparse approximation in the signal processing community seemed not really interested in the overcomplete dictionary learning problem. However, they often stated that one of the main advantage of this technique is the total freedom in designing the dictionary, which could be efficiently tailored to closely match signal structures. The first approach came from the neuroscience community. Olshausen and Field [59–61] started from the hypothesis that the response of the mammalian visual system to stimuli coming from natural images should be sparse. Sparsity is the criterion they used to model the efficiency of this *mammalian low bit rate image coder*. Their algorithm is representative of the probabilistic approach as they suppose that the coefficients c_i have a parameterized distribution with a peak at zero. All the algorithms of this class try to maximize a likelyhood function :

$$P(f_i \mid D) = \int P(c)P(f_i \mid c, D)dc. \tag{3.4}$$

The learning algorithms of the first class use different models for the distribution of the coefficients c_i. For example, [48, 49] use a Laplacian distribution for the coefficients. These algorithms generally start with a random dictionary which is optimized using an Expectation Maximization like procedure :

1. **Sparse Approximation.** For each training signal f_i find the coefficients c_i using the model dependant assumptions and the current dictionary.

2. **Dictionary Update.** Update the dictionary such that the mean approximation error of the training signals using the found coefficients c_i is minimized.

This procedure ends when the dictionary converged. This alternating optimization was first suggested as solving (3.4) is not feasible analytically in most cases.

The second class of dictionary learning algorithms uses generally another prior which is the similarity between the dictionary learning problem and clustering. Tropp [80] suggested that clusters can be considered as being atoms that provide a 1-term approximation of the data vectors. Whilst the sparse approximation problem uses a fixed dictionary, the clustering problem consists in finding the best dictionary to do 1-term approximation.

Vector Quantization has also a dictionary learning taste and this fact was pointed out many times [1–4, 27, 28, 43]. The close relationship that exists between dictionary learning and the famous *k-means* algorithm was more specifically exploited by Aharon et al. [1–4]. Their dictionary learning algorithm, named K-SVD, is a generalization of the *k-means* clustering algorithm.

The K-SVD algorithm aims to find the best possible dictionary regarding following minimization problem :

$$\min_{D,C}\{\|S - DC\|_F^2\} \text{ subject to } \forall i, \|c_i\|_0 < T_0, \tag{3.5}$$

where S are the training signals, C the sparse approximation coefficients and $\|.\|_F$ denotes the Frobenius norm.

The minimization is done iteratively by first fixing the dictionary and finding the sparsest coefficients matrix C. The choice of the approximation algorithm to use was left to the reader. The only constraint is to have an approximation with less than T_0 non zero terms. The second step consists in updating both the dictionary D and the non-zero entries of C. This is done very smartly by using the Singular Value Decomposition in such a way that sparsity is promoted.

The algorithm that we propose in this chapter is different from the ones presented above. It learns structured dictionaries i.e. the dictionary is computed from a smaller set of functions, called *generating functions*, that are subject to different transformations. More precisely, the considered transform is the translation and the dictionary is defined as :

$$\mathcal{D} = \{\{T_p g_k\}, k = 1 \dots K\} \tag{3.6}$$

where the dictionary is generated by applying all possible translations T_p to a set $\mathcal{G}$ of generating functions.

The use of this kind of dictionary is mainly motivated by two reasons. First, approximation algorithms may benefit from fast transforms that compute many projections at once for different translations. Using a Fast Fourier Transform permits to compute many correlations by a single multiplication in the transform domain. Second, a generating function may have a semantical meaning e.g. correspond to an object in an image or to a specific pattern in an audio track. In this case, it is of great interest to have access to the generating function and the undergone transformation separately. For example, the algorithm proposed by Olshausen and Field [61] learns many times the same pattern but at different positions on the support.

Different learning algorithms have been proposed to deal with shift invariant structures. In [71–73], Smith and Lewicki propose to represent a signal f by a sum of kernel functions that can be arbitrarily positioned in time. Blumensath and Davis also proposed a shift-invariant learning algorithm [8]. Both solutions differ in their underlying model from our solution; they use a convolutional model to generate the signals:

$$f = \sum_{k=1}^{K} e_k \star g_k + \epsilon. \tag{3.7}$$

where $\mathcal{G} = \{g_k\}_{k=1}^{K}$ is the set of generating functions, the vectors of position e_k are sparse and $\star$ denotes the convolution.

3.3 Principle and algorithm

Formally, the aim is to learn a collection $\mathcal{G} = \{g_k\}_{k=1}^{K}$ of real-valued generating functions g_k such that a highly redundant dictionary $\mathcal{D}$ adapted to a class of signals can be created by applying all possible translations to the generating functions of $\mathcal{G}$. The algorithm we propose differs from the previously presented methods as it does not try to minimize an objective function made of the approximation error and a sparsity measure of the coefficients. We look for generating functions that are able to represent well the features present in the signal. Sparsity is not directly introduced during the learning. Our hypothesis is that an overcomplete dictionary made of atoms that are able to well represent the features present in the signals mandatorily leads to sparse approximations.

Let T_p be the operator that translates an infinite signal by $p \in \mathbb{Z}$ samples. Let the set $\{T_p g_k\}$ contain all possible atoms generated by applying the translation operator to g_k. The dictionary generated by $\mathcal{G}$ is $\mathcal{D} = \{\{T_p g_k\}, k = 1 \dots K\}$. The learning is done using a training set $\{f_n\}_{n=1}^{N}$ of N training signals of infinite size and but non-zero only on their support of size S_f. Similarly, the size of the support of the generating functions to learn is S_g such that $S_g \leq S_f$.

Let define $\mathbf{g}_k \in \mathbb{R}^{S_g}$ the restriction of the infinite size signal g_k to its support. In an analog way, let define $\mathbf{f}_{n,p_n^{(i)}}$, the restriction of $T_{p_n^{(i)}} f_n$ to the support of g_k, of size S_g.

The proposed algorithm learns translation invariant generating functions iteratively. For the first one, the aim is to find g_1 such that the dictionary $\{T_p g_1\}$ is the most correlated in mean with the signals in the training set. Hence, it is equivalent to the following optimization problem:

$$\text{UP} : g_1 = \arg\max_{\|g\|_2=1} \sum_{n=1}^{N} \max_{p_n} a_{p_n,n} \mid \langle f_n, T_{p_n} g \rangle \mid^2, \tag{3.8}$$

where the normalization value $a_{p_n,n}$ is such that the learning signals restricted to the support of the generating function are of unit energy i.e.

$$a_{p_n,n} = \frac{1}{\|\mathbf{f}_{n,-p_n^{(i)}}\|_2}. \tag{3.9}$$

Different learning signals may have very different amplitudes. Normalization avoids giving more importance to one signal (or part of it) than to another.

For learning the next generating functions, some modifications have to be included to avoid learning functions that have already been learnt. Two options are feasible; modify the learning signals according to the previously learnt generating functions or modify the optimization problem. Choosing the first option is subject to a lot of uncertainty. Should the found generating functions be subtracted from the learning patches? If yes, where and at which position? In all signals? The second option is more elegant. The original optimization problem is modified to include a constraint penalizing a generating function if a similar one has already been found. Assuming that $k-1$ generating functions have been learnt, the optimization problem to find g_k can be written as:

$$\text{CP} : g_k = \arg\max_{\|g\|_2=1} \frac{\sum_{n=1}^{N} \max_{p_n} a_{p_n,n} \mid \langle f_n, T_{p_n} g \rangle \mid^2}{\sum_{l=0}^{k-1} \sum_p \mid \langle g_l, T_p g \rangle \mid^2}. \tag{3.10}$$

Finding the best solution to the unconstrained problem (UP) or the constrained problem (CP) is hard, and we propose to decompose it into two simpler steps that are alternately solved :

1. **Localize.** For a given generating function $g_k^{(i)}$, find the best translations $p_n^{(i)}$,

$$p_n^{(i)} = \arg\max_{p_n} a_{p_n,n} \mid \langle f_n, T_{p_n} g_k^{(i)} \rangle \mid^2 \tag{3.11}$$

2. **Learn.** Update $g_k^{(i+1)}$ by solving UP or CP, where the optimal translations p_n are fixed to the previously found values $p_n^{(i)}$.

The first step only consists in finding the location of the maximum correlation between each learning signal f_n and the generating function $g_k^{(i)}$. Let us now consider the second step. As the translation operator admits a well defined adjoint operator, $\langle f_n, T_{p_n} g_k \rangle$ can be replaced by $\langle T_{-p_n} f_n, g_k \rangle$. Let $\mathbf{F}^{(i)}$ be the $S_f \times N$ matrix, whose columns are made of

the signals f_n shifted by $-p_n^{(i)}$. More precisely, the j^{th} column of $\mathbf{F}^{(i)}$ is $a_{p_n,n}\mathbf{f}_{n,-p_n^{(i)}}$. We denote $\mathbf{A}^{(i)} = \mathbf{F}^{(i)}\mathbf{F}^{(i)^T}$.

With these notations, the second step, for the unconstrained problem (UP), can be written :

$$\mathbf{g}_k^{(i+1)} = \underset{||\mathbf{g}||_2=1}{\arg\max}\, \mathbf{g}^T\mathbf{A}^{(i)}\mathbf{g} \tag{3.12}$$

where $(.)^T$ denotes transposition. The best generating function $\mathbf{g}_k^{(i+1)}$ is the unit eigenvector associated with the largest eigenvalue of $\mathbf{A}^{(i)}$.

For the constrained problem, we want to force $g_k^{(i+1)}$ to be as decorrelated as possible from all the atoms in $\mathcal{D}_{k-1}$. This corresponds to minimizing

$$\sum_{l=1}^{k-1}\sum_{p} |\langle T_{-p}g_l, g\rangle|^2 \tag{3.13}$$

or, denoting

$$\mathbf{B}_k = \sum_{l=1}^{k-1}\sum_{p} \mathbf{g}_{l,-p}{\mathbf{g}_{l,-p}}^T, \tag{3.14}$$

to minimizing $\mathbf{g}^T\mathbf{B}_k\mathbf{g}$. With these notations, the constrained problem can be written :

$$\mathbf{g}_k^{(i+1)} = \underset{||\mathbf{g}||_2=1}{\arg\max}\, \frac{\mathbf{g}^T\mathbf{A}^{(i)}\mathbf{g}}{\mathbf{g}^T\mathbf{B}_k\mathbf{g}} \tag{3.15}$$

The best generating function $\mathbf{g}_k^{(i+1)}$ is the eigenvector associated to the biggest eigenvalue of the generalized eigenvalue problem defined in Eq. (3.15). Note that defining $\mathbf{B}_1 = \mathbf{Id}$, we can use CP for learning the first generating function $\mathbf{g}_1$.

The algorithm, which we call MoTIF, for Matching of Time Invariant Filters, is summarized in **Algorithm 3.3**.

Algorithm 3 Principle of the learning algorithm (MoTIF)

$k = 0$, training signals set $\{f_n\}$
while not enough generating functions **do**
  $k \leftarrow k + 1$, $i \leftarrow 0$
  $\mathbf{B}_k \leftarrow \sum_{l=1}^{k-1}\sum_p \mathbf{g}_{l,-p}{\mathbf{g}_{l,-p}}^T$
  while no convergence reached **do**
    $i \leftarrow i + 1$
    for each f_n, find $p_n^{(i)} = \arg\max_p a_{p_n,n} \mid \langle f_n, T_p g^{(i)}\rangle \mid$, by locating the maximum correlation between f_n and $g^{(i)}$,
    $\mathbf{A}^{(i)} \leftarrow \sum_{n=1}^{N} a_{p_n,n}^2 \mathbf{f}_{n,-p_n^{(i)}}{\mathbf{f}_{n,-p_n^{(i)}}}^T$
    find $\mathbf{g}_k^{(i+1)} = \arg\max_{||\mathbf{g}||_2=1} \frac{\mathbf{g}^T\mathbf{A}^{(i)}\mathbf{g}}{\mathbf{g}^T\mathbf{B}_k\mathbf{g}}$, that is the eigenvector associated to the biggest eigenvalue of the generalized eigenvalue problem $\mathbf{A}^{(i)}\mathbf{g} = \lambda\mathbf{B}_k\mathbf{g}$.
  end while
end while

For the first generating function, the MoTIF algorithm converges in a finite number of iterations to a function locally maximizing the unconstrained problem (3.8). Proving that the algorithm converges in this case is trivial. Let assume that if there are different possible positions for the maxima and that one of these corresponds to the maxima of the previous step then this one is chosen i.e. if

$$a_{p_n^{(i-1)},n} \mid \langle f_n, T_{p_n^{(i-1)}} g^{(i)} \rangle \mid = \max_p a_{p,n} \mid \langle f_n, T_p g^{(i)} \rangle \mid \tag{3.16}$$

then $p_n^{(i)} = p_n^{(i-1)}$. Ensuring that this requirement is met is trivial when implementing the algorithm. Finding the best positions as well as updating the atom are operations that tend to maximize the objective function $\sum_{n=1}^{N} a_{p_n,n} \mid \langle f_n, T_{p_n} g \rangle \mid^2$. The hypothesis that we made ensures that if no better positions are found at step i then the same ones as before are chosen which implies that $\mathbf{g}_k^{(i+1)} = \mathbf{g}_k^{(i)}$. Additionally, the number of possible combinations of positions is huge but finite, implying that the algorithm converges in a finite number of iterations. This convergence property does not depend on the initialization.

We do not claim that the algorithm converges to the global minimum but we claim that it converges to a local one. Figure 3.1 presents 10 runs of MoTIF for finding the first generating function using the learning set made of the images used by Olshausen [61]. The last column contains the final results. They have the same appearance but are all different. The other columns contain the generating functions during the 11 first iterations of MoTIF. It starts from random initializations. The convergence to a waveform near the final result is fast and confirms the validity of the proposed method. Next to the finial result, we present the number of iterations MoTIF needed to converge as well as the value of the objective function.

Our implementation of the algorithm includes an additional constraint; we want to center the energy on the support. To do so, at each iteration, we compute the center of mass of the previously found generating function and subtract it from the geometric center of the support. This *shifting* vector is added to all positions found at the current step. The positions are rounded to the nearest possible value. These values corresponds to pixels of images or to samples of music for example. Obviously, in this setting, the nice property of convergence is no longer guaranteed to hold true. However, experimental results show that it does. Shifting adds *innovation* that has not been taken into account by the search phase and thus, changes the value of the objective function in an unexpected way. Figure 3.2 shows the evolution of the value of the objective function for three experiments presented by figure 3.1. The value of the objective function corresponding to Experiment 6 is decreasing at iteration 7. The lower part of the figure confirms that at this point the new generating function is very different from the previous one. By attentively looking at the generating function in figure 3.1, it is clear that a shift to the left happened. A shift also happened to the 7^{th} generating function at iteration 4; however, in this case, the objective function did not decrease.

Figure 3.1 also makes clear that all the solutions found by MoTIF to the unconstrained problem are very close each other. The constraint avoids finding only one kind of generating

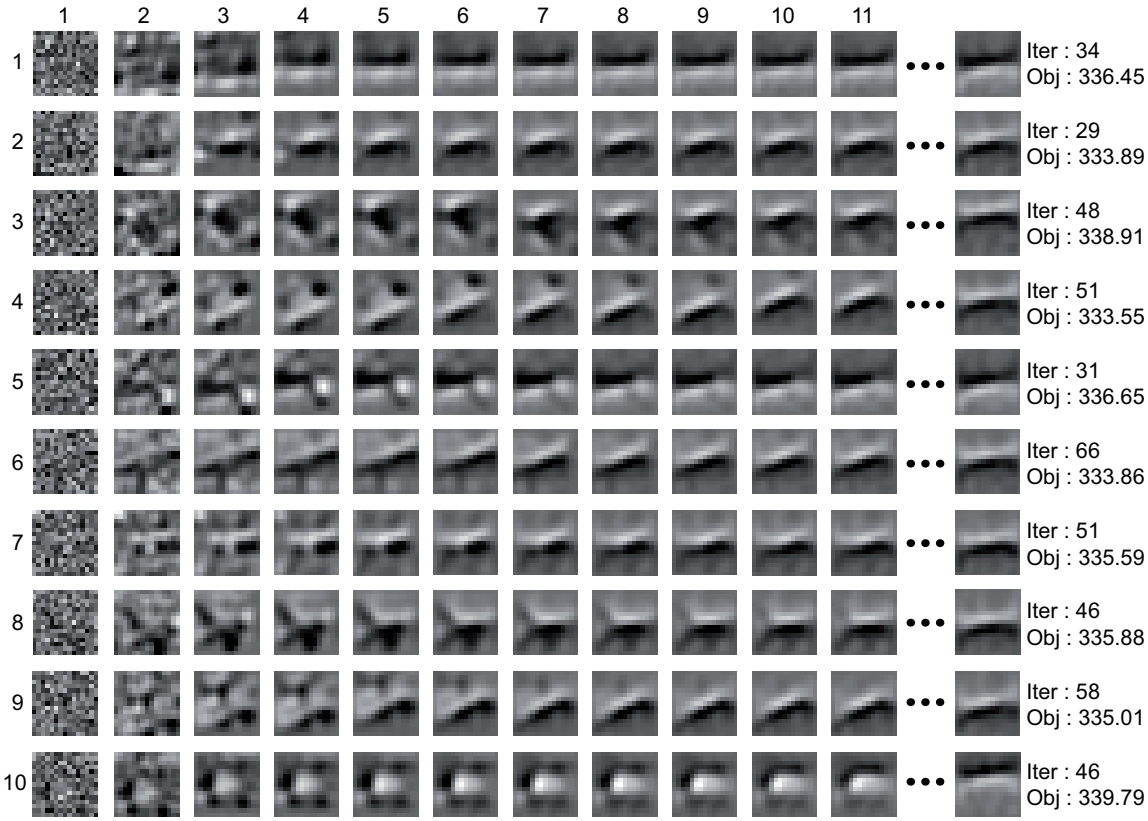


Figure 3.1 – *First generating function learnt by MoTIF* 10 *times.*

function by forcing the new ones to be different from the previous ones. Using constraints avoids altering the training signals. On the other hand, some generating functions satisfy better the decorrelation condition than the adequation to the training signals.

Figure 3.3 presents 100 generating functions of size 15×15 that have been learnt by MoTIF using the same dataset as for the previous examples. They are spatially localized and oriented. They are oscillating in directions different from the orientations, at different frequencies. We recognize some familiar shapes that may make us think of Gabor atoms, line edge detectors and curved edge detectors. The two first categories were already observed in [7] and the third ones complete the range of natural features. Learning curved edge detector is a major contribution of this algorithm in the field of learning visual shapes.

The first generating function is very similar to the multiple runs presented in Figure 3.1. The others are learnt by adding the decorrelation constraint to the original problem. The generating functions learnt just after the first one are mainly high frequency due to the added constraint ensuring that they are different from the first one.

The objective function linked to the unconstrained problem measuring the adequation may be expressed as :

$$\sum_{n=1}^{N} \max_{p_n} a_{p_n,n} \mid \langle f_n, T_{p_n} g \rangle \mid^2 . \tag{3.17}$$

It gives an indication of how well g fits the learning signals.

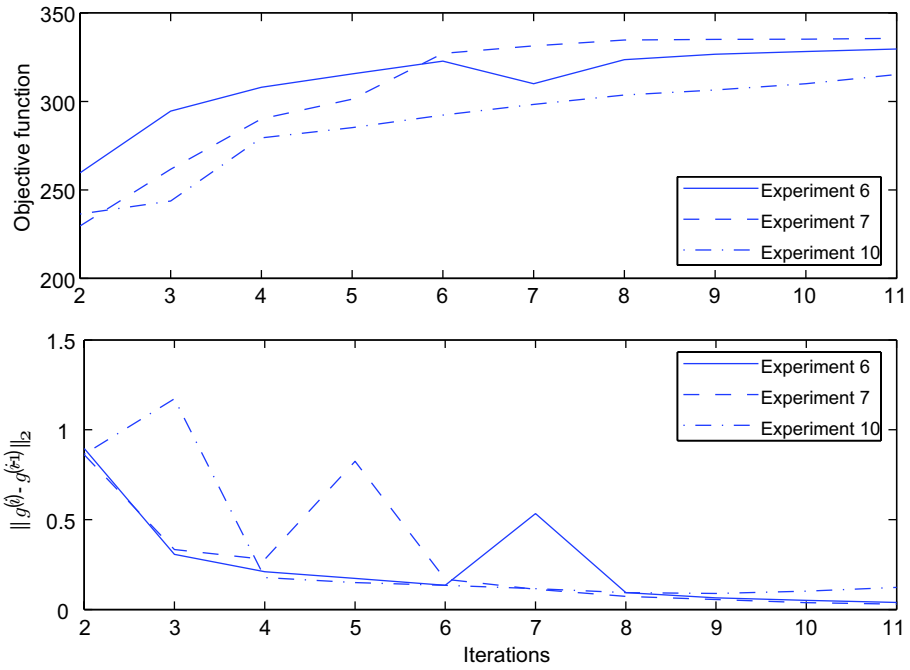


Figure 3.2 – *Evolution of the value of the objective function for experiment 6, 7 and 10 of the generating functions presented in figure 3.1 (top). The l_2-norm of the difference to the generating function from the last iteration is also presented (bottom).*

Figure 3.4 presents the values of the objective function (3.17) for all learnt generating functions of Figure 3.3. The ones satisfying too much the constraint are clearly identified as they exhibit smaller values. This is roughly the case for the second up to the tenth generating function.

The evolution of the waveforms also depends on the decorrelation constraint. For example, in Figure 3.5, the second generating function is mostly determined at the second step. It is an additional indication that it is more satisfying the decorrelation constraint than fitting the learning data.

3.4 Experiments

3.4.1 Synthetic experiments

The first experiment consists in exploring the ability of the algorithm to recover correctly a set $\mathcal{G}^O = \{g_k^O\}_{k=1}^K$ of known generating functions referred to as the original set of functions. Starting from this set, a sparse coefficient vector c is randomly created. It defines a signal :

$$s = \sum_{k=0}^{N-1} c_k \phi_k, \quad \phi_k \in \mathcal{D} = \{\{T_p g_k^O\}, k = 1..K\}.$$

The training set $\{f_n\}$ is obtained by taking the maximal number of non overlapping parts of the signal s. The size of the patches f_n is such that $supp(f_n) = 2 * supp(g_k^O) - 1$, where

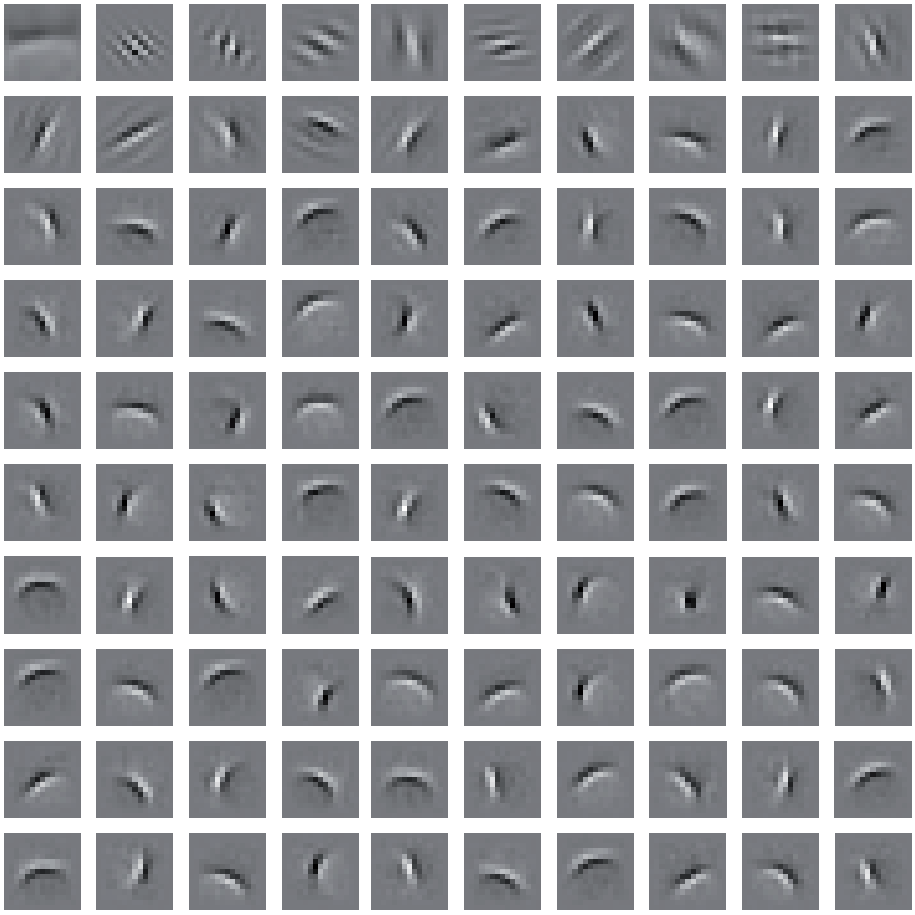


Figure 3.3 – *Hundred generating functions learnt on natural images.*

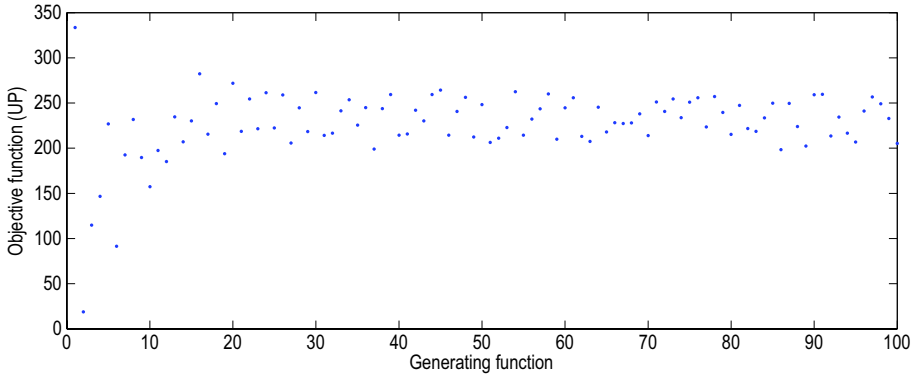


Figure 3.4 – *Values of the unconstrained objective function* (3.17) *for the generating function presented in Figure 3.3.*

supp denotes the size of the support. These patches are used by the MoTIF algorithm to learn a set $\mathcal{G}$ of translation invariant generating functions. A function g_i^O from the original set $\mathcal{G}^O$ is said to be recovered if $\max_{g \in \mathcal{G}} |\langle g_i, g \rangle| > \delta$.

We created 3000 original sets of generating functions made of 3 Gabor atoms with random normalized frequency between 0 and 0.5. The size of their spatial support is 16.

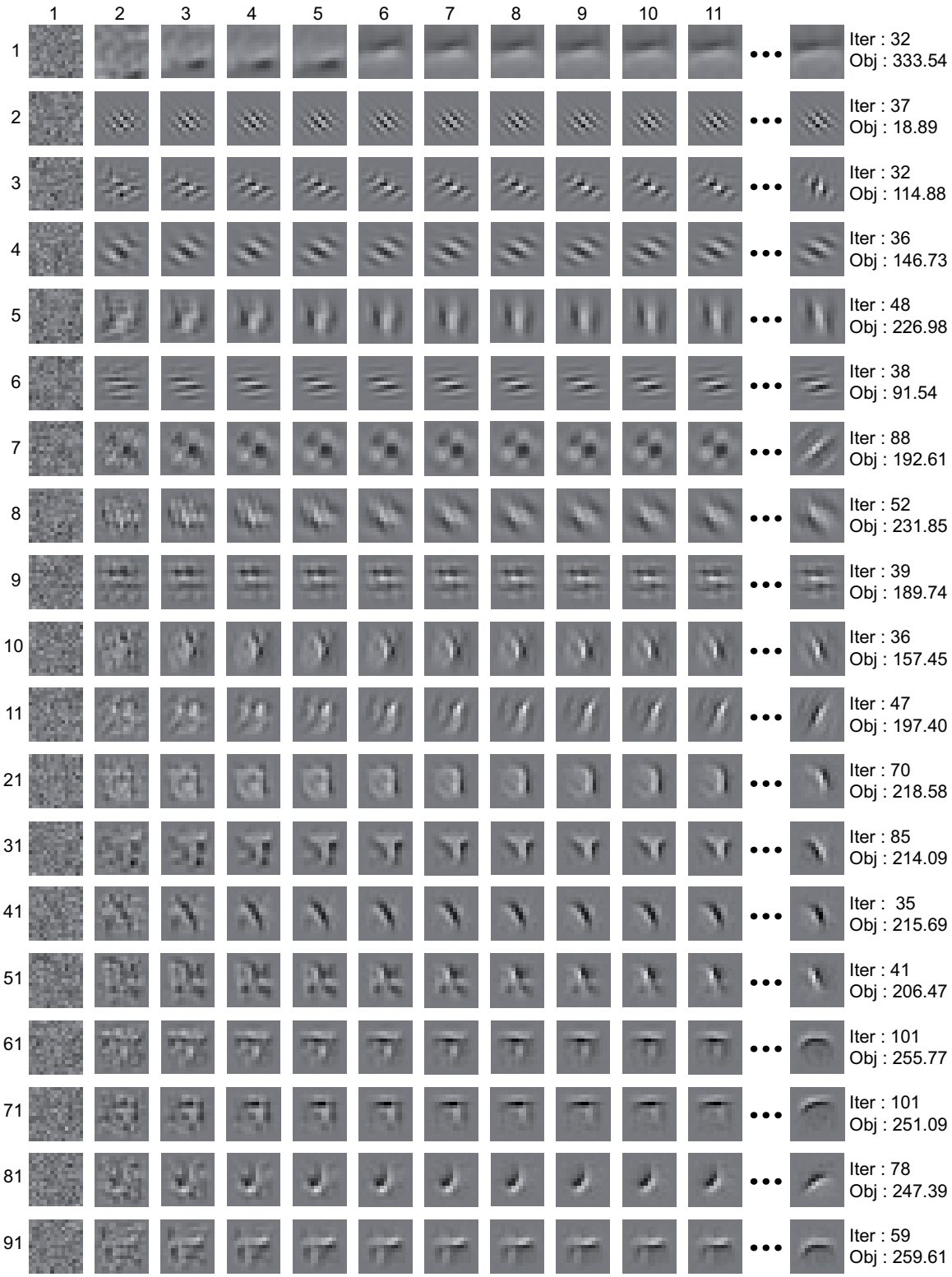


Figure 3.5 – *First iterations of some generating functions presented in Figure 3.3.*

Each of these generating functions was present 10 times in a signal of size 1600 with a random amplitude between 0 and 1. The number of patches f_n used was 298. For each set

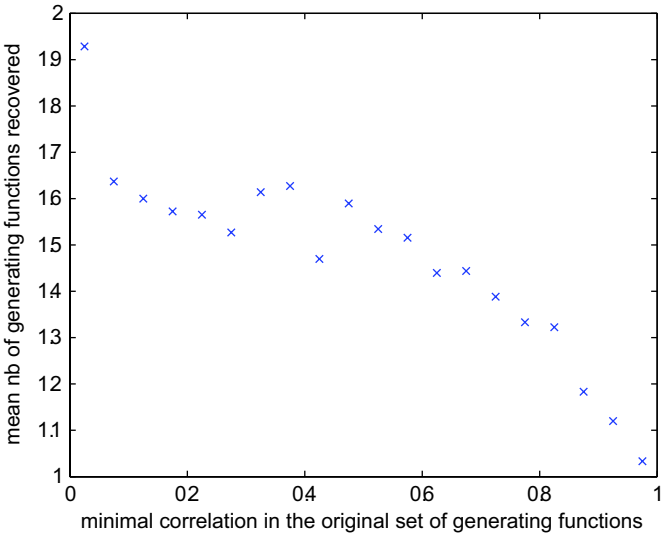


Figure 3.6 – *Mean number of recovered generating functions as a function of the minimal coherence of the reference dictionary.*

of generating function, we run the algorithm 10 times on 10 different signals.

Figure 3.6 illustrates the recovery ability of the MoTIF algorithm. It presents the mean number of generating functions recovered as a function of the minimal correlation of the original set $\mathcal{G}^O$ computed as $\min_{i,j} \max_p |\langle T_p g_i^O, g_j^O \rangle|$, which means that the correlation between other atoms can only be higher. The equivalence limit δ for two generating functions was fixed to 0.8.

For the same settings, in more than 2 cases out of 3, the first generating function found by MoTIF is one from the original set. To recover the next atoms, the constrained optimization problem (CP) has to be solved. Thus, the next functions are constrained to be as uncorrelated as possible with the past found functions, which is clearly not the case when the original set of functions is highly coherent. This leads to a poor rate of recovery when the minimal coherence is higher than 0.6. Recovering is easier when dealing with rather uncorrelated set of functions. Indeed, for very small values of the minimal correlation, in mean, nearly two functions out of three are recovered. In between these two extreme cases (minimal coherence between 0.2 and 0.6), the algorithm's behavior is rather constant and recovers more than half of the functions.

3.4.2 Experiments with natural signals

The second experiment studies the ability of a dictionary learnt on real data to sparsely approximate signal of the class of the learning data, compared to a classical dictionary made of Gabor multi-scale generating functions. The class of signals we consider is music. More precisely, one song; ninety percent of the signal has been used for learning whilst the second part is kept to test the approximation ability.

The reference dictionary that we are using is made of 150 Gabor functions. A simple

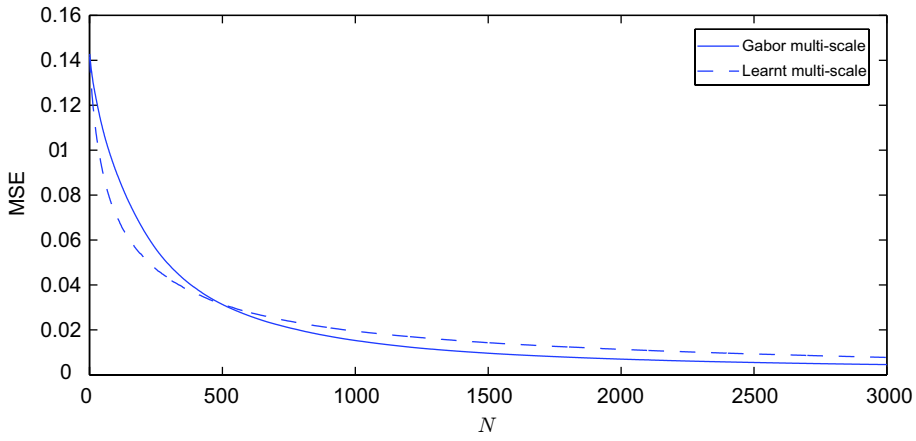


Figure 3.7 – *Approximation abilities of a learnt set of generating functions regarding a dictionary multi-scale Gabor atoms.*

optimization was applied to this dictionary; the frequencies were chosen to be in the frequency range of the song i.e. we used 50 different normalized frequencies spread between 0 and 0.25. Three different scales where used for the generating functions. The size of the support of these generating functions is 1024. The gabor dictionary $\mathcal{D}_G$ is generated by applying all possible transaction to the set of generating functions $\mathcal{G}_G$.

We decided to use MoTIF to obtain a dictionary that has the same flavor as the reference one. We learnt three dictionaries of 50 generating functions each with different supports: 256, 512 and 1024. For each run of MoTIF, we used the maximal information from the signal i.e the learning signals was divided into respectively 4086, 2042 and 1020 training patches. The three sets of generating functions are concatenated to form $\mathcal{G}_L$ which defines the learnt dictionary $\mathcal{D}_L$ that has multi-scale abilities.

To compare the approximation performances of both dictionaries, we used Matching Pursuit [51] to approximate a test signal of 45000 samples from the same song. The test signal has been approximated with 3000 terms. We reconstructed the approximations of the signals using N terms and computed the mean square error. The results are presented by Figure 3.7. At the beginning, the mean square error of the signal approximated using the learnt dictionary decreases much faster than with the Gabor atoms. The learnt dictionary is adapted to the considered signal and contains meaningful features present in the test signal.

Figure 3.8 presents the generating functions that have most been used in both decompositions. The right column contains Gabor generating functions that have rather a small scale. This may be an indication that this dictionary did not best suit the signal. The learnt generating functions are less precisely located in frequency. However, their scale is bigger than for the Gabor dictionary; we may deduce that this dictionary better suits the testing data.

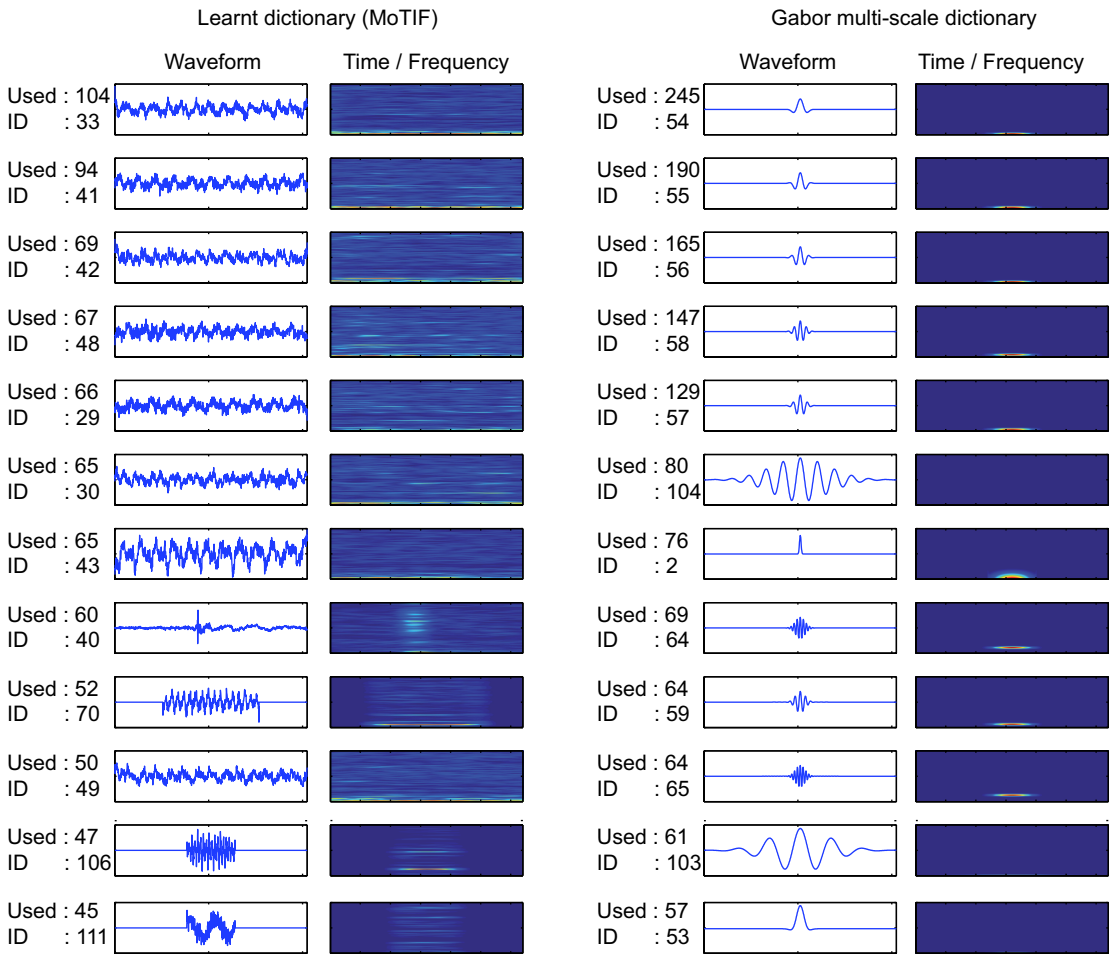


Figure 3.8 – *Comparison of the most used generating functions for approximation.*

3.5 Extensions

The simplicity of this algorithm is an advantage in many situations as it allows to identify meaningful structures at reasonable computational costs.

One of our hypothesis is that creating such a dictionary intrinsically holds the ability to sparsely approximate signals from the same class. Lesage and Grivonval advocate that not including explicitly a sparsity constraint implies that there is no guarantee that the dictionaries learnt by MoTIF lead to sparse approximations. Starting from this point, they propose to adapt the K-SVD algorithm to match the invariance constraints [47] in an analog manner as MoTIF does. They also present a structure where dictionaries are created by applying linear transforms to generating functions; translation is a particular case. Remember that K-SVD is an iterative algorithm made of a *sparse approximation* phase and a *dictionary update* phase. It has the particularity that the second phase goes along with a refinement of the approximation coefficients. Lesage showed that using his model of dictionary, the second step may be reduced to a problem that is very similar to

what we presented :

$$g_k^{opt} = \underset{\substack{g \\ \|g\|_2^2==1}}{\arg\max}\, g^T A(g) g. \tag{3.18}$$

where the matrix $A(g)$ is symmetric, square and is of same length than the generating functions. If the matrix $A(g)$ is not depending on the generating function g then the solution of this problem is the eigenvector of A associated to the eigenvalue having the biggest amplitude.

In the same context, Mailhé and Gribonval propose to update the generating functions whilst the parameters of the linear transform function are found at the *sparse approximation* step. Additionally, they suppose that the linear transformation preserve the norm and that two atoms created from the same generating function are orthogonal to each other if the parameters of the transformation are different. Their final result is that the optimal generating function is as follows :

$$g_k^{opt} = \frac{\sum i_k = 1^{I_k} F^*_{\alpha_{i_k}}(s)}{\| \sum i_k = 1^{I_k} F^*_{\alpha_{i_k}}(s) \|_2} \tag{3.19}$$

where F^*_α denotes the adjoint of the linear transformation F_α.

These examples tried to improve the results of MoTIF by generalizing the possible transforms under which to seek for invariant generating functions. They also added explicitly a constraint of sparsity by using the learning paradigm proposed by Aharon and Elad, the K-SVD algorithm.

Let us now look at a further example extending MoFIF. Monaci proposed to use it for multichannel data and in particular for multi-modal data [54–56]. For this kind of data, synchrony is a major issue and is generally tightly linked to the underlying physic. For example, a video sequence of a speaking person contains sound that is generally synchronized with the motion of the lips. The sampling of different modalities may be very different; typical values of the sampling rates are 8 kHz for audio signals and 29.97 frames per second for the video.

A multi-modal generating function g_k consist of an arbitrary number M of modalities. For simplicity and clarity reasons, Monaci considered the bimodal case $M = 2$. A bimodal generating function is expressed as $g_k = (g_k^{(a)}, g_k^{(v)})$ where one can think of $g_k^{(a)}$ as an audio modality and $g_k^{(v)}$ as a video modality of audiovisual data. These components are not homogeneous in dimensionality; however, they share a common temporal dimension. Monaci defined a translation operator $\mathcal{T}_p = (\mathcal{T}_p^{(a)}, \mathcal{T}_p^{(v)})$ that acts on the different modalities taking into account the different sampling rates. This temporal translation is homogeneous across channels and thus preserves synchrony which is very important when dealing with multi-modal signals.

Using the same formalism as before, it is possible to redefine the unconstrained problem for the multi-modal case as follows :

$$\text{UPmm} : g_1 = \underset{\|g^{(a)}\|_2 = \|g^{(v)}\|_2 = 1}{\arg\max} \sum_{n=1}^{N} \max_{p_n} \sum_m |\, \langle f_n^{(m)}, \mathcal{T}_{p_n}^{(m)} g^{(m)} \rangle \,|^2 , \tag{3.20}$$

which has to be solved simultaneously for all modalities. The constrained problem for the multi-modal case is change in an analog way. Solving UPmm or CPmm is generally more complex than the corresponding problems in the standard setting. A MoTIF like approach is a good solution for finding solutions to these problems. Instead of a recursive two steps approach, an iterative four steps one has been proposed :

1. ***Localize :*** for a given audio generating function $g_k^{(a),(j-1)}$ found at iteration $j-1$, find the best translations $p_n^{(a)(j)}$ on the audio modality.
2. ***Learn :*** update $g_k^{(v),(j)}$ by solving UPmm or CPmm only for the video modality, with the translations fixed to the values $p_n = p_n^{(a),(j)}$ found at step 1. These values have to be transformed into the modality according to the different sampling rates.
3. ***Localize :*** find the best translations $p_n^{(v),(j)}$ using the function $g_k^{(v),(j)}$.
4. ***Learn :*** update $g_k^{(a),(j)}$ by solving UPmm or CPmm only for the audio modality, with the translations fixed to the values $p_n^{(v),(j)}$ found at step 3.

The temporal synchrony between generating functions on the two modalities is enforced at the learning steps (2 and 4), where the optimal translations found for one modality are also kept for the other one. This procedure is illustrated by Figure 3.9 (Figure 6.1 of [54]).

Monaci demonstrated the ability of the proposed algorithm in this case the two modalities (audio and video), sharing a common temporal axis. This experiment has been presented in his thesis [54] and we show the results here to show a practical application of MoTIF.

The joined audiovisual dictionary is learned on a training set of four clips representing the mouth of four different persons pronouncing the digits from zero to nine in English. The audio was recorded at 44 kHz and sub-sampled to 8 kHz, while the gray-scale video was recorded at 29.97 fps and at a resolution of 70×110 pixels. The total length of the training sequences is 1140 video frames, i.e. approximately 38 seconds. The video sequences are "whitened" using a filter that equalizes the variance of the input sequences in all directions as suggested in [61]. The whitened sequences are then low-pass filtered to remove the high-frequency artifacts typical of digital video signals.

The learning is performed on audio-video patches $(f_n^{(a)}, f_n^{(v)})$ extracted from the original signals. The size of the audio patches $f_n^{(a)}$ is 6407 audio samples, while the size of the video patches $f_n^{(v)}$ is 31×31 pixels in space and 23 frames in time. We learn 20 generating functions g_k consisting of an audio component $g_k^{(a)}$ of 3204 samples and a video component $g_k^{(v)}$ of size 16×16 pixels in space and 12 frames in time.

Figure 3.10 presents the 20 learnt multi-modal generating functions. For the video part, the frames are shown separately. The shape that appear are very similar to the results presented in section 3.3. One may recognize spatially localized and oriented edge detector functions. Looking globally at the video sequences, we remark that the edges are moving. These movements represent the movements of different parts of the mouth.

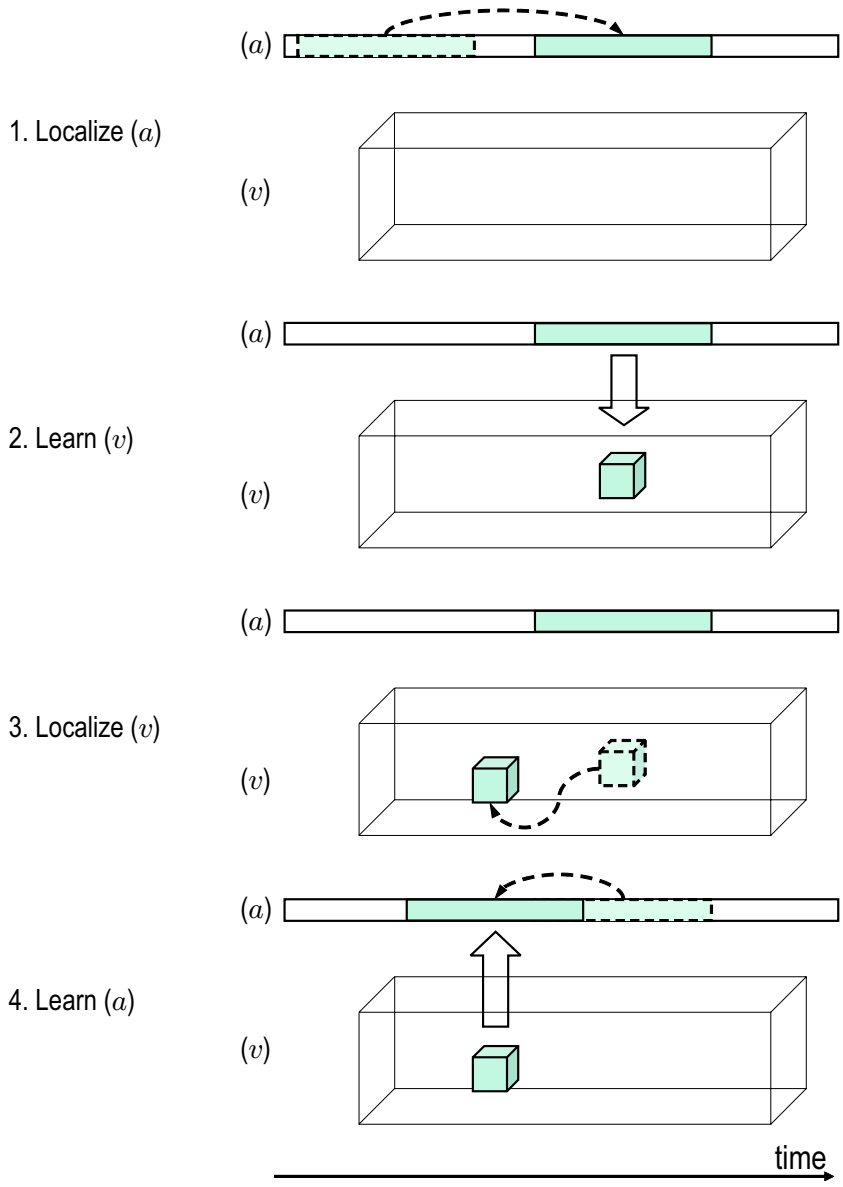


Figure 3.9 – *Schematic representation of the multi-modal version of MoTIF.* Step 1 *: using the available generating function for modality* (a)*, find the best translations in* (a)*.* Step 2 *: using the found translations on* (a)*, update the generating function in* (v)*.* Step 3 *: using this generating function, find the best translations for modality* (v)*.* Step 4 *: using the translations found in modality* (v)*, update the generating function in* (a)*.*

The audio parts of the generating functions contain almost all the numbers present in the training sequences. Thus, the learning algorithm captures well high-level signal structures representing the synchronous presence of meaningful acoustic and visual patterns. The simplicity of the MoTIF algorithm permitted to rapidly adapt the algorithm to the highly complex multi-modal case. On the other hand, its efficiency allowed for finding meaningful results out of a huge amount of data.

3.6 Discussion

We have presented a new method for learning a set of translation invariant functions adapted to a class of signals. At every iteration, the algorithm produces the waveform that is the most present in the signals and adds all its shifted versions to the dictionary. A constraint in the objective function forces the learnt waveforms to have low correlation, such that no atom is picked several times. The main drawback of this method is the fact that the few generating functions following the first one are mainly due to the decorrelation constraint, more than the attachment to the signal. Despite this, the constrained algorithm seems to capture the underlying processes quite well, notably when they are really decorrelated. The learnt dictionaries show ability to sparse decompose the corresponding signals. On real data like images, the learnt generating functions are edge detectors (spatially local and oriented) as previously found by Bell and Sejnowski. Some extensions of this algorithm are presented, as learning multi-modal atoms on multi-modal signals. Using this type of learning, some applications in multichannel source separation can be expected. Another extension, based on the properties of the inner product, is to replace the translation invariance by the invariance to a whole set of transformations that admit a well defined adjoint (e.g. translations + rotations for images).

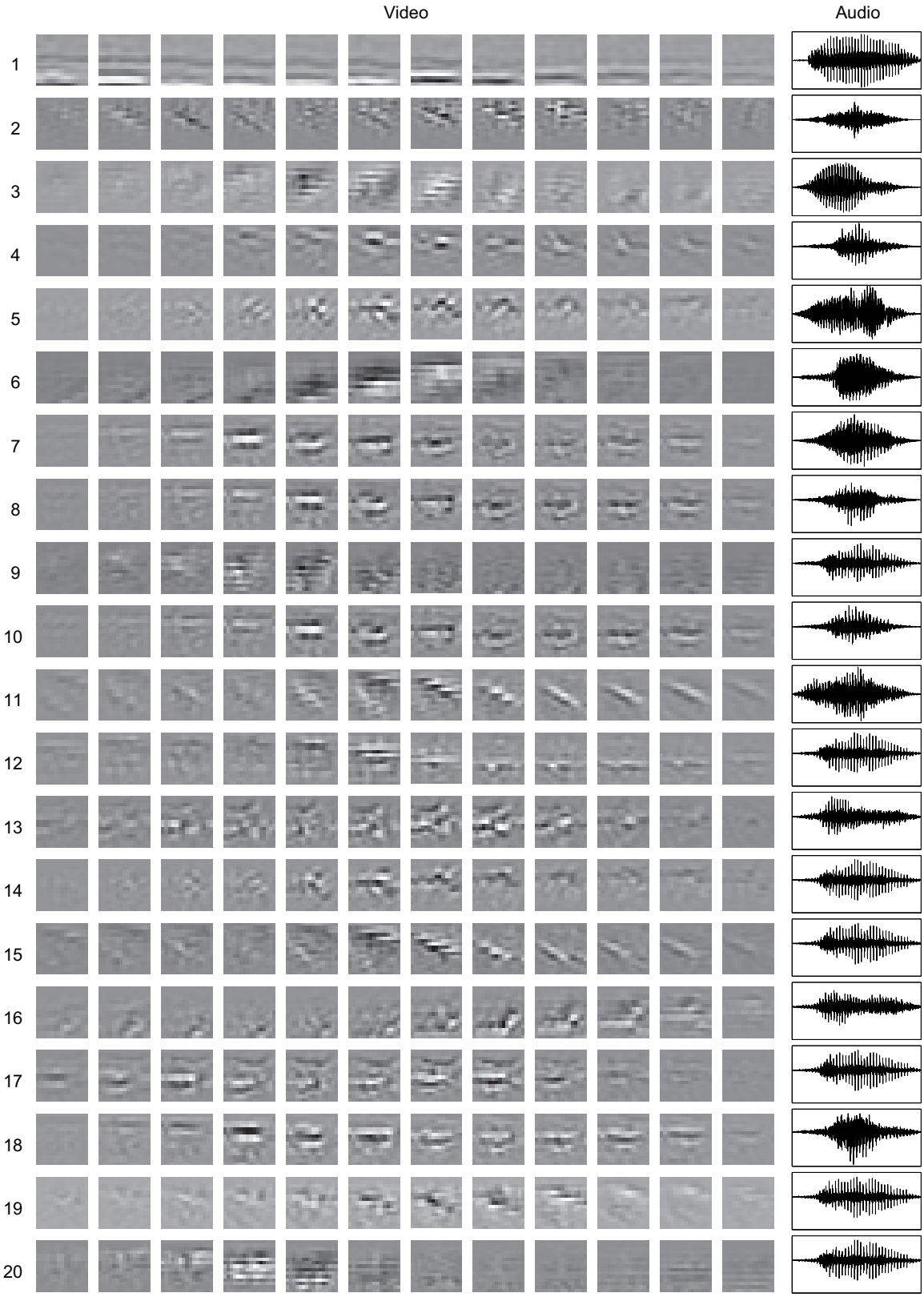


Figure 3.10 – *Audio-video generating functions of* Dictionary 2. *Twenty learned functions are shown, each consisting on an audio and a video component. Video components are on the left, with time proceeding left to right. Audio components are on the right, with time on the horizontal axis.*

Finding Nearest Neighbors in a Set of Compressible Signals

4

4.1 Motivations

Numerous applications demand that we manipulate large sets of very high-dimensional signals. A simple yet common example is the problem of finding those signals in a database that are closest to a query. In this chapter, we tackle this problem by restricting our attention to a special class of signals that have a sparse approximation over a basis or a redundant dictionary. We take advantage of sparsity to approximate quickly the distance between the query and all elements of the database. In this way, we are able to prune recursively all elements that do not match the query, while providing bounds on the true distance. Validation of this technique on synthetic and real data sets confirms that it could be very well suited to process queries over large databases of compressed signals, avoiding most of the burden of decoding.

Chapter 2 dealt with the problem of computing sparse approximation and chapter 3 dealt with problem of having *efficient* dictionaries to represent a given class of signals. In this chapter, we assume that we are given sparse approximations of signals and we will ignore *how* they have been computed. We will however require that our approximants possess a particular structure. Suppose first that the terms of the approximation

$$\tilde{f}_N = \sum_{k=0}^{N-1} c_k g_k, \quad g_k \in \mathcal{D}, \quad \|f - \tilde{f}_N\|_2 \leq \epsilon\,. \tag{4.1}$$

are re-ordered in decreasing order of magnitude, i.e such that $| c_0 | \geq | c_1 | \geq \cdots \geq | c_{N-1} |$. Strict sparsity requires that the number of non-zero coefficients N be small. However we can slightly relax this definition by asking that the magnitude of the coefficients drops quickly to very small values such that there are only few big coefficients. A signal that is well

approximated by such an expansion over a dictionary is termed *compressible*, highlighting the idea that most of the information is contained in few coefficients [24]. Usually, and that is the case in this chapter, the sorted coefficients are assumed to follow a power-law decay; the i^{th} largest is such that:

$$| c_i | \leq \mathrm{C}\ i^{-\gamma}. \tag{4.2}$$

for $\gamma \geq 1$ and some positive constant C. The decay parameter γ may depend on both the signal and dictionary.

Dictionaries used to define the class of compressible signals need not be redundant. Piece-wise smooth signals for example are compressible on wavelet bases and that characteristic is at the heart of the good performances of wavelets for compression or denoising [50]. In many cases however, a simple basis cannot efficiently capture signal characteristics within a compressible representation. Dictionaries offer more flexibility and dramatically enlarge the class of compressible signals.

In this chapter, we explore how sparsity can be used to handle huge amount of data at a lower cost. More precisely, we tackle the problem of computing in an efficient manner the correlation of a single query signal with a huge set of compressible signals. Our algorithm uses only the components c_k, g_k of the signal model (4.1), hence can be seen as working in the *transform domain*. Since compression is key in storing large collections of signals, we thus potentially avoid the extra burden of having to decode large amounts of data for searching or browsing through the database.

In section 4.3.1 we derive the scalar product of two signals according to the parameters of their respective sparse representation. It allows to bound parts of the scalar product when dealing with *compressible* signals. In section 4.3.2 we present an algorithm to compute efficiently the projection of a signal on a set of signals. The algorithm uses the bound previously introduced. Section 4.3.3 and 4.4 present improved versions of the simplest bound. Section 4.5 presents different experiments to illustrate the different bounds as well as the algorithm itself. We conclude in section 4.6 on the benefits of this new approach and list the perspectives we will consider.

4.2 Introduction

With the advent of digital cameras and portable music players, modern digital signal processing has also to face the challenge of voluminous databases of signals. Clearly, signal processing algorithms must be adapted to problems where each user manipulates large collections of signals. Finding the nearest neighbor in a database is fundamental for many applications; [44] presents a good overview of this field. Generally, when the data is lying in a high dimensional space, a dimensionality reduction step is used to lower the complexity of the query. In the field of signal processing, the dimensionality reduction resulting from the sparsity of an approximation has been exploited for different tasks such as analysis, de-noising or compression. Roughly speaking, the sparser the representation, the better it

is for applications.

Before moving on to the core of this chapter, let us briefly describe how our contributions can be compared with existing techniques. The field of nearest-neighbor algorithms is very wide and still extremely active, we thus certainly couldn't hope to provide here a fair survey. However, we would like to highlight some key results and orientations and this will also allow us to specify constraints used in our framework.

Finding the nearest neighbor of a query in a set $\mathcal{F}$ of I d-dimensional vectors can be solved by brute force with $\mathcal{O}(dI)$ operations. Clearly, when I is big (and that is the case in most applications), this could be prohibitive. A lot of work has been devoted to trying to reduce the amount of computations needed to deal with large data sets. Most of the recent approaches have a cost scaling like $\mathcal{O}(\exp d \log I)$ provided the data base is first pre-processed to create an efficient data structure [6, 14, 15, 22]. It has to be noted that a computational cost exponential in d does not improve on the brute force technique when d is large enough, i.e $d > \log I$, highlighting the so called *curse of dimensionality*. Various algorithms have been proposed to solve this problem, with complexity that roughly scales in $\mathcal{O}(d^{\beta}\text{polylog}(dI))$, for some $\beta > 1$ and an appropriate data structure [38, 42, 44].

This short survey brings us to our main constraint. In this chapter, we target applications in user centric multimedia databases, i.e images, audio that reside on the user's computer, and in this setting we cannot afford large preprocessing time. More particularly we must be able to add and remove entries in the database at no cost. We don't extract low dimensional feature vectors from our signals. Instead we use sparse representations both for compression and description of the data. Our data structure is thus simple and forced upon us: the description of each item in terms of the coefficients and atoms' indexes in (4.1). As for how sparsity N depends on the dimension d, it is hard to give a precise rule. Though N is much smaller than d, we will assume that it scales linearly with d. We thus have high-dimensional vectors in our database.

4.3 A simple deterministic algorithm

4.3.1 Notations and warm-up

For the sake of generality, we will work from now on with a dictionary, i.e a collection of unit L^2 norm atoms $g_i \in \mathbb{R}^d$, $i = 1, ..., L$, with L possibly much larger than d. The dictionary is often represented as a matrix D whose columns are the atoms g_i. Using this formalism, $G = D^*D$ is the Gram matrix of the dictionary and contains all possible scalar products between atoms, i.e. $G_{ij} = \langle g_i \mid g_j \rangle$.

Let us consider a set of compressible signals $\mathcal{F} = \{f^i\}_{i=i}^I$:

$$f^i = \sum_{j=1}^{N} c_j^i g_{k_j^i}. \tag{4.3}$$

where $g_{k_i} \in \mathcal{D}$. The vector $\mathbf{k^i}$ contains the indices of the atoms in the dictionary and is such that the projections c_j^i are in decreasing order of magnitude. Note that we have voluntarily

discarded the N-term approximation error in (4.3), and we will keep on doing so from now on. We will discuss later the influence of this term.

The aim of this chapter is to provide an efficient method to find, in the set of signals $\mathcal{F}$, the one that is closest to a query signal $q = \sum_{l=1}^{N_q} b_l g_{k_l}$ that is also compressible with N_q terms. The magnitudes of the projections are also decreasing with l. The scalar product $\langle f^i \mid q \rangle$ between a signal from the set and the new one can be written as follows:

$$\begin{aligned} \langle f^i \mid q \rangle &= \langle \sum_{j=1}^{N} c_j^i g_{k_j^i} \mid \sum_{l=1}^{N_q} b_l g_{k_l} \rangle, \\ &= \sum_{j=1}^{N} \sum_{l=1}^{N_q} c_j^i b_l G_{k_j^i, k_l}. \end{aligned} \tag{4.4}$$

where $G_{k_j^i, k_l} = \langle g_{k_j^i} \mid g_{k_l} \rangle$ is an entry of the Gram matrix of the dictionary $\mathcal{D}$.

The aim of the algorithm is to exploit sparsity, i.e N_q, $N \ll d$, in order to find the best matching signal. It is done by eliminating rapidly the signals whose scalar products with the query is too small. To do so, we rewrite the scalar product presented in eq. (4.4) as follows:

$$\langle f^i \mid q \rangle = \sum_{k=2}^{N+N_q} s_k^i, \tag{4.5}$$

where s_k^i represents the part of the scalar product coming from atoms participating in both decompositions such that the sum of j and l is equal to k. For the i^{th} signal of the set $\mathcal{F}$, it corresponds to:

$$s_k^i = \sum_{\substack{j,l \\ j+l=k \\ j \leq N, l \leq N_q}} c_{k_j^i} b_{k_l} G_{k_j^i, k_l}. \tag{4.6}$$

The signals of the set $\mathcal{F}$ and the query q are compressible. According to eq. (4.2), there exists γ and a constant C such that $\mid c_j^i \mid \leq C j^{-\gamma}$ and $\mid b_l \mid \leq C l^{-\gamma}$. Since the entries of the Gram matrix are between -1 and 1, it is possible to bound the magnitude of s_k^i as follows:

$$\mid s_k^i \mid \leq \sum_{\substack{j,l \\ j+l=k \\ j \leq N, l \leq N_q}} C^2 j^{-\gamma} l^{-\gamma} \tag{4.7}$$

4.3.2 Iterative candidate rejection

Computing the scalar product of two discrete d-dimensional signals requires d multiplications and $d-1$ additions. Let us now suppose that these signals have an exact-sparse representation using respectively N and N_q terms and that the entries of the Gram matrix G have been pre-computed. Computing the scalar product using eq. (4.4) needs $3N * N_q$

multiplications and $N * N_q - 1$ additions. If $N = N_q$, there is a computational gain only if $N < \sqrt{\frac{d}{2}}$, so for very sparse signals. However, using the formalism described in the previous section, it is possible to compute iteratively the scalar product of two signals that have a sparse and compressible representation. When searching for the best matching signal in a huge set $\mathcal{F}$, it is of great interest to be able to eliminate in an early stage the signals that have no chance to match. If the scalar product is computed in an iterative way, our aim is to eliminate signals by estimating at each step an upper and a lower bound on the final scalar product, which is possible by using the bound presented by eq. (4.7). To do so, let us first define:

$$S_K^i = \sum_{k=2}^{K} s_k^i, \tag{4.8}$$

which represents the part of the scalar product $\langle f_i \mid q \rangle$ found by taking into account the atoms whose sum of indices is smaller or equal to K. Using the same formalism, it is possible to express the missing part of the scalar product:

$$R_K^i = \sum_{k=K+1}^{N+N_q} s_k^i. \tag{4.9}$$

If we had kept track of the approximation error in our initial model (4.3), we would have to add it to this residual. We simply assume that this error is sufficiently smaller than the typical values of R_K^i we will be working with. If the signals are well-compressible, this will be the case and this is indeed what our simulations suggest. Using the two preceding equations, let us express the scalar product as $\langle f^i \mid q \rangle = S_K^i + R_K^i$, $\forall K$, $2 \leq K \leq N + N_q$. The value of S_K^i can be computed iteratively as $S_K^i = S_{K-1}^i + s_K^i$.

When looking for the signal that is most correlated with the query, one computes the absolute value of the scalar product, disregarding the sign of the projection. Thus, $\forall K$, $2 \leq K \leq N + N_q$ the following relation holds:

$$\mid S_K^i \mid - \mid R_K^i \mid \leq \mid \langle f_i \mid q \rangle \mid \leq \mid S_K^i \mid + \mid R_K^i \mid . \tag{4.10}$$

Using eq. (4.7), it is possible to upper bound the residual part of the correlation $\mid R_K^i \mid$.

$$\mid R_K^i \mid \leq \sum_{k=K+1}^{N+N_q} \mid s_k^i \mid \leq C^2 \sum_{k=K+1}^{N+N_q} c_{k,N,N_q} (\frac{k^2}{4})^{-\gamma} = \tilde{R_K^i}, \tag{4.11}$$

where c_{k,N,N_q} is the number of possible products between atoms such that the sum of their indices is equal to k and knowing that we have N terms for f_i and N_q terms for the query:

$$c_{k,N,N_q} = \begin{cases} k-1 & \text{if } 2 \leq k \leq N_q + 1; \\ N_q & \text{if } N_q + 1 < k \leq N; \\ N + N_q - k + 1 & \text{if } N < k \leq N + N_q; \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

Using the bound $\tilde{R_K^i}$ defined in eq. (4.11) it is possible to upper and lower bound the correlation at any iteration K as follows :

$$m_K^i \leq \mid \langle f_i \mid q \rangle \mid \leq M_K^i, \tag{4.13}$$

where $m_K^i = \mid S_K^i \mid -\tilde{R_K^i}$ and $M_K^i = \mid S_K^i \mid +\tilde{R_K^i}$ are respectively the lower and the upper bound. It is obvious that if $\exists K$ s.t. $M_K^j < m_K^i$ then $\mid \langle f_i \mid q \rangle \mid > \mid \langle f_j \mid q \rangle \mid$. This principle is illustrated by Figure 4.1 where the maximal value some candidates could eventually reach is lower than the worst case of the best matching candidate. The pseudo-code illustrating the proposed algorithm is presented by table 4.3.2.

Algorithm 4 Find best matching signal in a database of exact-sparse N-terms signals

INPUTS: A signal $q = \sum_{i=1}^{N_q} a_i g_i$, $g_i \in \mathcal{D}$.
A set of signals $\mathcal{F}$ having a exact-sparse representation using N terms.
The Gram matrix G of the dictionary used to represent the signals.
OUTPUT: $min_i \mid \langle f_i \mid q \rangle \mid$, the index of the signal that best matches q.
INITIALIZATION: $P = \{i\}_{i=1}^{\|\mathcal{F}\|}$ the indices of the signals in the set $\mathcal{F}$.
$K = 2$.
$S_1^i = 0$, $\forall i$.
while card$(P) > 1$ **do**
  Compute all $S_K^i = S_{K-1}^i + s_K^i$.
  Compute all m_K^i and M_K^i.
  $S = \{f_i\}_{M_K^i < \max_i m_K^i}$
  $P = P \setminus S$.
  $K = K + 1$.
end while

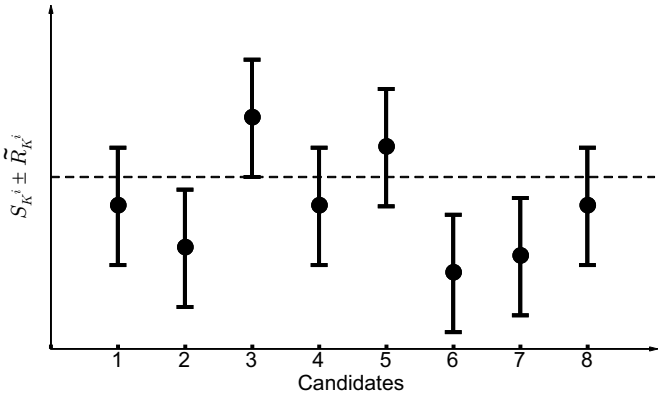


Figure 4.1 – *Elimination of candidates 2,6 and 7.*

4.3.3 Improved bounds

The absolute value of the projection is not decreasing at uniform rate. The slope is much steeper at the beginning than at the end. This behavior is closely related to the redundancy of the dictionary. Suppose now that the slope is bounded as $\mid a_i \mid \leq Ci^{-\gamma_i}$ i.e. for each projection a_i, we learn a local parameter instead of using a global constant value for γ. It is now possible to bound $\mid s_k^i \mid$ as follows:

$$\mid s_k^i \mid \leq \sum_{\substack{i,j \\ i+j=k \\ i \leq N, j \leq N_q}} C^2 i^{-\gamma_i} j^{-\gamma_j}. \tag{4.14}$$

One reason for locally adjusting the decay parameter γ is that greedy algorithms for example tend to produce compressible approximants that have a non-monotonic decay : usually γ is much bigger for the first few coefficients and then tends to decrease slowly.

Let $\{f_n\}$ be a huge collection of signals having an exact-sparse representation using N terms. The projection of the i^{th} atom of the decomposition of f_j is denoted by a_i^j. Let also have the maximal i^{th} projection $A_i = \max_j \mid a_i^j \mid$. The best parameters C and γ used previously can be found by solving:

$$\{C, \gamma\} = \arg\min_{C,\gamma} \sum_{i=1}^{N} (A_i - Ci^{-\gamma})^2, Ci^{-\gamma} \geq A_i, \forall i. \tag{4.15}$$

Using the C we have learnt, we define the values γ_i for the improved bound as follows :

$$\gamma_i = -\log_i \frac{A_i}{C}. \tag{4.16}$$

Using the locally optimized values better matches the behavior of the absolute values of the projections. The bound is used in the algorithm to eliminate the worst matching candidates. Thus, improving the quality of this bound directly acts on the quality of the results of the proposed algorithm. It has to be noticed that the values of the bound do not depend on which signal the algorithm is working with and it can thus be computed offline.

4.4 A probabilistic approach

The worst case bounds presented in the previous sections are based on the hypothesis that the signs of the scalar products between atoms conspire against us and the entries of the Gram matrix of the dictionary are always equal to 1. These worst case hypotheses are far from typical cases. It is straightforward to see that if the signs are positive or negative with equal probability, then the sequences s_k^i (for sufficiently big values of k) would be zero mean. Results in the field of concentration of measure show that functions defined on a large probability space most of the time take values that do not fall too far away from the average case. We will now use these techniques to obtain much sharper bounds for R_K^i.

We will have to accept that these bounds hold with high probability and not with absolute certainty.

Without loss of generality, we assume that the coefficients are always positive. The dictionary could simply be augmented to contain also all *opposite* atoms so that this property holds. At each step of the algorithm, we estimate the following amplitude:

$$\begin{aligned} \mid R_K^i \mid &= \mid \sum_{k=K+1}^{2N} s_k^i \mid & (4.17) \\ &= \mid \sum_{\substack{j,l \\ j+l \geq K+1 \\ j \leq N, l \leq N}} c_j^i b_l G_{k_j^i, k_l} \mid . & (4.18) \end{aligned}$$

First, let suppose that the *worst* case is met for the values of the Gram matrix but that the corresponding signs are random, $+1$ or -1 with probability $\frac{1}{2}$. From these considerations, we rewrite the previous equation as follows:

$$\begin{aligned} \mid R_K^i \mid &= \mid \sum_{k=K+1}^{2N} s_k^i \mid & (4.19) \\ &= \mid \sum_{\substack{j,l \\ j+l \geq K+1 \\ j \leq N, l \leq N}} \epsilon_{j,l} c_j^i b_l \mid & (4.20) \\ &= \mid \sum_n \epsilon_n a_n \mid . & (4.21) \end{aligned}$$

where $\sum_n \epsilon_n$ is a Rademacher sequence i.e. ϵ_i is $+1$ or -1 with equal probability.

Theorem 2. *Let* $\mathbf{a}$ *be a real vector and* ϵ *a Rademacher sequence. Then* $\forall t > 0$

$$P(\mid \sum_n \epsilon_n a_n \mid > t) \leq 2e^{-\frac{1}{2}t^2/\|\mathbf{a}\|_2^2}. \quad (4.22)$$

The proof of Theorem 2 can be found in chapter 4 of [45]. Using this theorem, it is straightforward to see that

$$P(\mid \sum_n \epsilon_n a_n \mid \leq t) \geq 1 - 2e^{-\frac{1}{2}t^2/\|\mathbf{a}\|_2^2}. \quad (4.23)$$

Since the magnitude of the coefficients are bounded, the entries of $\mathbf{a}$ are also bounded and this gives us a simple upper bound of its l_2-norm. It is then easy to find an upper bound $\tilde{R_K^i}$ for a given probability p by solving $1 - 2e^{-\frac{1}{2}(\tilde{R_K^i})^2/\|\mathbf{a}\|_2^2} = p$. This bound will be discussed in our experiments (Section 4.5) for different values of p. Note that $\tilde{R_K^i}$ is influenced in a unfavorable way by the l_2-norm of $\mathbf{a}$. On the other hand, this reasoning remains general enough to be valid for any dictionary $\mathcal{D}$; in particular for orthogonal bases.

The hypothesis that the vector containing the coefficients signs is a Rademacher sequence is certainly more reasonable than the worst case. However, we still have a levee

to pull. So far indeed, we supposed that the entries of the Gram matrix are always 1 and that is very far from reality. We will now investigate how much we can squeeze from a probabilistic model, assuming that the entries of the Gram matrix are drawn at random and that the projections are always positive as before. The residual R_K^i can be rewritten as follows:

$$\begin{aligned} \mid R_K^i \mid &= \mid \sum_{k=K+1}^{2N} s_k^i \mid & (4.24) \\ &= \mid \sum_n G_n a_n \mid . & (4.25) \end{aligned}$$

where G_n are chosen as independent random variables modelling the entries of the Gram matrix of the dictionary. Let us now turn to the choice of these variables. By definition, all values of the Gram matrix are in the interval $[-1, 1]$. It is hard to deduce any general rule. For redundant dictionaries used in practice an histogram of the entries of the Gram matrix has a peak around zero signaling weakly correlated atoms, and a dirac at one representing the values on the diagonal. Without more exploitable structure, we choose the simplest random variables, i.e uniform random variables. This choice will lead to pessimistic bounds, but note that since (4.25) models R_K^i as a large sum of these variables we will be able to easily evaluate and control the resulting distribution.

Let $U(n)$ be a symmetric uniform random variable. Its associated probability density function is:

$$f_{U(n)}(x) = \begin{cases} \frac{1}{2n} & \text{if } -n \leq x \leq n; \\ 0 & \text{otherwise.} \end{cases} \quad (4.26)$$

Eq. (4.25) can be rewritten as a sum of not identically distributed independent uniform random variables:

$$\begin{aligned} \mid R_K^i \mid &= \mid \sum_n a_n U_n(1) \mid & (4.27) \\ &= \mid \sum_n U_n(a_n) \mid & (4.28) \end{aligned}$$

The probability distribution of uniformly distributed random variables with different symmetrical ranges is presented in [53]. For a more general approach and a survey of different results, please refer to [9]. The probability density function of a sum of uniform random variables $\sum_{n=1}^m U_n(a_n)$ is given by Theorem 1 of [9] and in [53], the cumulative density function is given as follows:

$$F_m(x) = \sum_{\overrightarrow{\epsilon} \in \{0,1\}^m} (-1)^{\sum_{i=1}^m \epsilon_i} (\frac{1}{2}(x + \sum_{i=1}^m a_i - \prod_{i=1}^m \epsilon_i a_i))^m \Big/ m! \prod_{i=1}^m a_i. \quad (4.29)$$

$F_m(x)$ is combinatorial and becomes difficult to evaluate in presence of a large number m of random variables. An asymptotic density distribution $\hat{f}_m(x)$ exists and is presented in [53]. For small values of K, the number m of uniform random variables in the sum is large and represents a typical case for using this approximation :

$$\hat{f}_m(x) \approx \left[\frac{3}{2\pi \sum_{i=1}^m a_i^2}\right]^{\frac{1}{2}} \exp\left[-\frac{3}{2}\frac{x^2}{\sum_{i=1}^m a_i^2}\right]. \quad (4.30)$$

Since this asymptotic probability density function is gaussian, the associated asymptotic cumulative density function is a simple error function :

$$\hat{F}_m(x) \approx \frac{1}{2}\Big(1 + \mathrm{Erf}\big(\sqrt{\frac{3}{2}}\frac{x}{\|\mathbf{a}\|_2}\big)\Big). \quad (4.31)$$

Given a probability parameter p, we simply have to find the bound $\tilde{R}_K^i$ such that $F_m(\tilde{R}_K^i) - F_m(-\tilde{R}_K^i) = p$.

4.5 Experiments

In this section, we evaluate our rejection techniques on simulated and real data. First, we consider the aspects related to the estimation of the different bounds. Second, we show their influence on the behavior of the algorithm. Finally, we present two experiments of finding the closest image in a set of compressible images.

In order to illustrate the different properties and behaviors of the algorithm, we constructed a simple synthetic dataset as follows. A set of 7200 images from the COIL-100 database [58] normalized to have the same energy have been approximated using matching pursuit with a dictionary made of anisotropic atoms [30]. This yields compressible approximants, with nicely decaying projection coefficients. We then synthesized signals by multiplying these coefficients with atoms randomly selected in a dictionary made of the union of the Haar and DCT bases. The choice to take projections values coming from real approximations is motivated by the fact that the bounds mostly rely on them. Our experiments will thus be representative of the behavior of the algorithm in real cases.

The non probabilistic upper bounds do not depend on the dictionary that is chosen but only on the projections, on the number of terms N and N_q of the sparse representations and on the amount of atoms that have already been taken into account by the algorithm. Figure 4.2 presents the values γ_i bounding the amplitudes of the projections. These values have been learnt from the projections and this figure illustrates well the fact that the amplitude

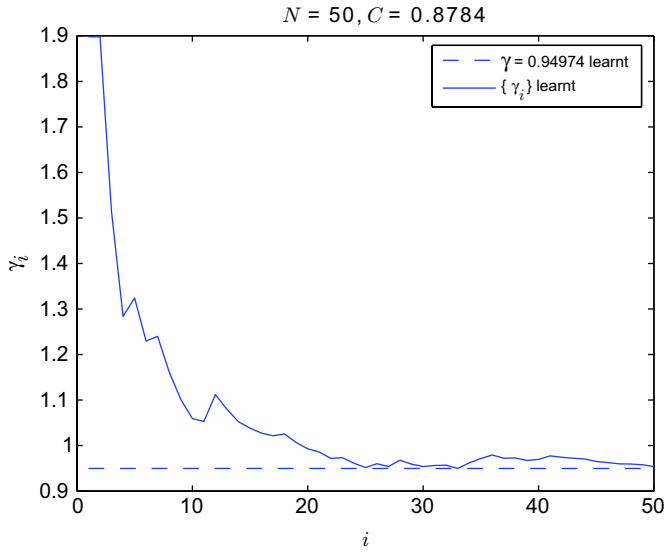


Figure 4.2 – *Locally optimized* $\{\gamma_i\}$ *compared to overall learnt* γ.

of the projections is not decreasing at uniform rate. The slope is steeper at the beginning than at the end.

Figure 4.3 exhibits the values of the different bounds presented in this chapter. The simple bound is far from reality especially for small values of K whilst the bound using the learnt values γ_i (shown in Figure 4.2) is almost half. This fact is illustrated by the upper plot. The two other plots present probabilistic approaches. In the middle, the signs come from a Rademacher sequence whilst on the lower part, the entries of the Gram matrix are modelled as uniform random variables. Varying the value of the parameter p leads to different bounds. For both probabilistic models, when $p = 1$ the obtained curve would be the bound obtained using the learnt values γ_i on the upper part of the figure. Using small values for p leads to sharper bounds at the cost of increasing the probability that the algorithm commits errors whilst still guaranteeing that $P(\text{error}) \leq 1 - p$. Notice that, even when p is large (i.e $p = 0.9$), the associated bound is an order of magnitude smaller than in the deterministic case.

The computational gain of the algorithm greatly depends on its ability to eliminate at early stages non suitable candidates. Figure 4.4 illustrates the relation between the number of potential candidates and the different bounds. Most of the energy of the signals is caught by the first terms of the decompositions. Signals whose first atoms have low correlation with the first atoms of the query signal are not likely to be the best ones and should rapidly be eliminated by the algorithm. This favorable case happens if the bound is tight enough. The upper part of Figure 4.4 presents the evolution of the cardinality of the set of potential candidates during the execution of the algorithm for the simple bound and the one based on the learnt values γ_i. For these bounds, the elimination of non suitable candidates happens late during the execution of the algorithm. Clearly, the probabilistic bounds are much tighter and this has a direct influence on the behavior of the algorithm. Indeed, one can observe on the other plots of Figure 4.4 that the cardinality of the set of

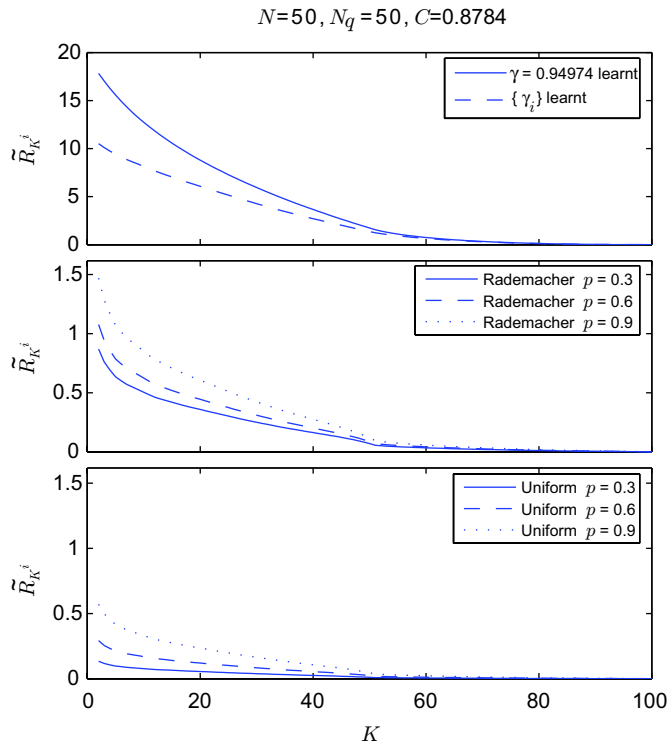


Figure 4.3 – *Comparison of the different bounds. The simplest bound is greatly improved by using a better upper bound for the energy (up). The signs modeled as a Rademacher sequence (middle). The entries of the Gram matrix modeled by a uniform random variable (down).*

potential candidates is decreasing very quickly in earlier stages. The choice of the value of p has a major influence on the behavior of the algorithm and on its ability to eliminate non suitable candidates given a query signal.

At step K of the algorithm, the computational complexity depends on the number signals in the set of potential candidates. The evolution of the cardinality of this set is closely linked to the chosen bound. By extension, the overall complexity of the algorithm also depends on the chosen bound. When using a probabilistic approach, it is possible that the algorithm commits mistakes and eliminates the best matching signal. Note though that our models are *pessimistic* and the true probability of error is smaller than the corresponding $1-p$. Figure 4.5 (middle-top) shows the relationship between the probability parameter p of the used probability model and the computed probability of error. The computations were done for p going from 0.01 to 0.99 by steps of 0.01. According to the model, the probability of error should be lower than $1-p$ which is the case for all experiments. It has to be noticed that the *real* probability of error is much lower than the theoretic bound of the model. For example, the algorithm committed no error for values of p bigger than 0.43. However, we can not derive any general rule as these results depends on the characteristics of the dictionary.

During its execution, the algorithm considers successively pairs of atoms. For each of them, it has to multiply the corresponding entry of the gram matrix by both projection

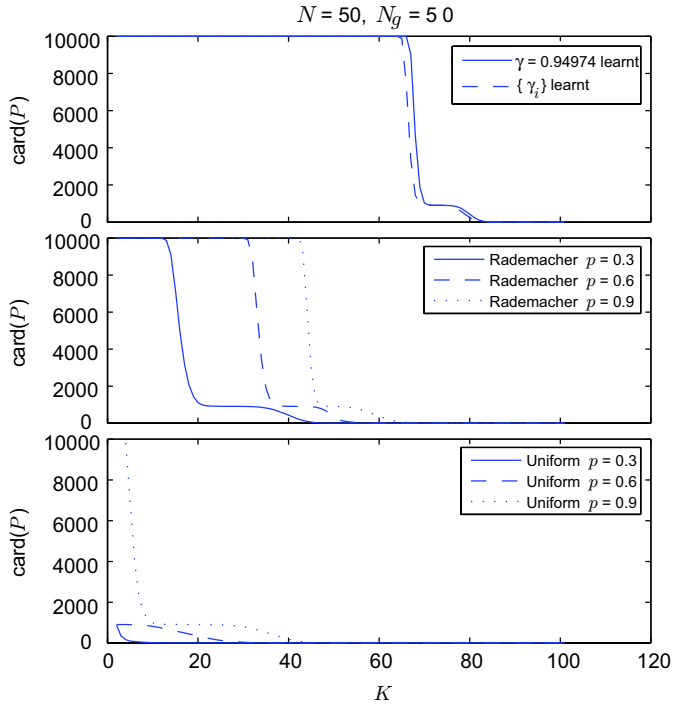


Figure 4.4 – *Impact of the different bounds on the number of candidates. A database of 3000 signals of 2000 samples is used. The three parts of the figure present the cardinality card(P) of the set of potential candidates for the different bounds defined in the chapter averaged over 20000 queries.*

parameters and add it to the current estimation of the scalar product. We considered that each of these individual operation has unit weight. Thus, for each pair, the algorithm makes 3 operations. We did not take into account the computations needed to find the candidates to eliminate. Figure 4.5 (top) presents the number of operations needed for different values of p. The direct computation of the scalar product would require 2000 multiplications and 1999 addition for each signal in the database. Thus, the overall number of operations is roughly 40×10^6.

When computing nearest neighbors in high dimensional spaces, one often seeks for a $(1+\epsilon)$-approximate nearest neighbor $f_i \in \mathcal{F}$ of q such that $\forall f_k \in \mathcal{F}$, $\text{dist}(f_i, q) \leq (1+\epsilon)\text{dist}(f_k, q)$. Where dist() is some distance function or some norm. Our algorithm computes scalar products between signals, we have thus used the simplest distance:

$$\text{dist}(f_i, q) = \left(1 - \left(\frac{|\langle f_i, q\rangle|}{\|f_i\|_2 \|q\|_2}\right)^2\right)^{1/2}. \tag{4.32}$$

Since all signals have roughly unit norm, $\text{dist}(f_i, q) \approx \sqrt{1 - |\langle f_i, q\rangle|^2}$. Note that one could use more complicated distances, for example kernel distances, provided they are still based on evaluating scalar products. The traditional kernels used in classification are based on scalar products between vectors. Some of them are listed below, but the interested reader

can get more insight in any textbook, for example [17] :

$$\begin{aligned} \text{Polynomial kernel:} \quad & k(f_i, q) = \langle f_i, q \rangle^{\alpha} \\ \text{Radial basis function:} \quad & k(f_i, q) = \exp -\frac{\|f_i - q\|^2}{2\sigma^2} \\ \text{Sigmoid:} \quad & k(f_i, q) = \tanh\left(\kappa \langle f_i, q \rangle + c\right) \end{aligned}$$

Choosing the best distance, though, is application dependent and out of the scope of the present chapter. We will thus perform all our experiments using the simple euclidean distance. Note as well that some classification schemes, most notably support vector machines [17], rely heavily on selecting particular data points based on their scalar products with reference vectors. These techniques could potentially benefit from fast rejection algorithms.

For each experiment, we computed the value ϵ, called slackness. The two lower parts of Figure 4.5 presents the mean and the maximal values of the slackness. The mean slackness is very near 0; this is due to the fact that when the algorithm correctly identifies the best signal, the corresponding ϵ is 0. Generally, one is more interested in the maximal possible value for ϵ as it bounds the worst case. For small values of p, the maximal value of the slackness is quiet high whilst it decreases rapidly when p increases.

Since no conditions were imposed on the dictionary apart from the fact that the signals should be compressible, it is straightforward to see that the bound using the Rademacher sum is valid whatever dictionary is used. In the present experiment, a set of 7200 images from the COIL-100 database [58] where approximated with 200 terms of a wavelet decomposition (i.e the dictionary is an orthogonal basis). The images are of size 128×128 and the filter used for the wavelet transform is a Daubechies of length 20. The database of signals is made of 2500 randomly chosen images and the other ones where used to test the algorithm. Figure 4.6 shows that the cardinality of the set of potential signals decays quickly with the number of iterations. Different parameters p have been used to obtain the bound, but this didn't change significantly the behaviour of the algorithm. Moreover, it has to be noticed that the algorithm always found the best signal in the database.

For our next set of experiments, we turn to a redundant dictionary. Let the set $\mathcal{F}$ be a collection of images having a sparse decomposition using a dictionary $\mathcal{D}$. We used images from the ORL face database [68].The dictionary is built using generating functions that are scaled, rotated and translated [30]. The generating function is made of a Gaussian in one direction and its second derivative in the other direction. It has a good ability to capture edges and is well located in space and frequency.

In our experiments, the atoms have translation parameters that take any positive integer value smaller than the size of the image. The rotation parameter varies by increments of $\frac{\pi}{18}$. The scaling parameters are uniformly distributed on a logarithmic scale from one up to an eighth of the size of the image, with a resolution of one third of octave. The scaling along the second derivative part is always smaller. The dictionary also contains Gaussian atoms. Their translation parameters can take the same values as for the anisotropic atoms, their scaling is isotropic and varies from $\frac{1}{32}$ to $\frac{1}{4}$ of the size of the image on a logarithmic

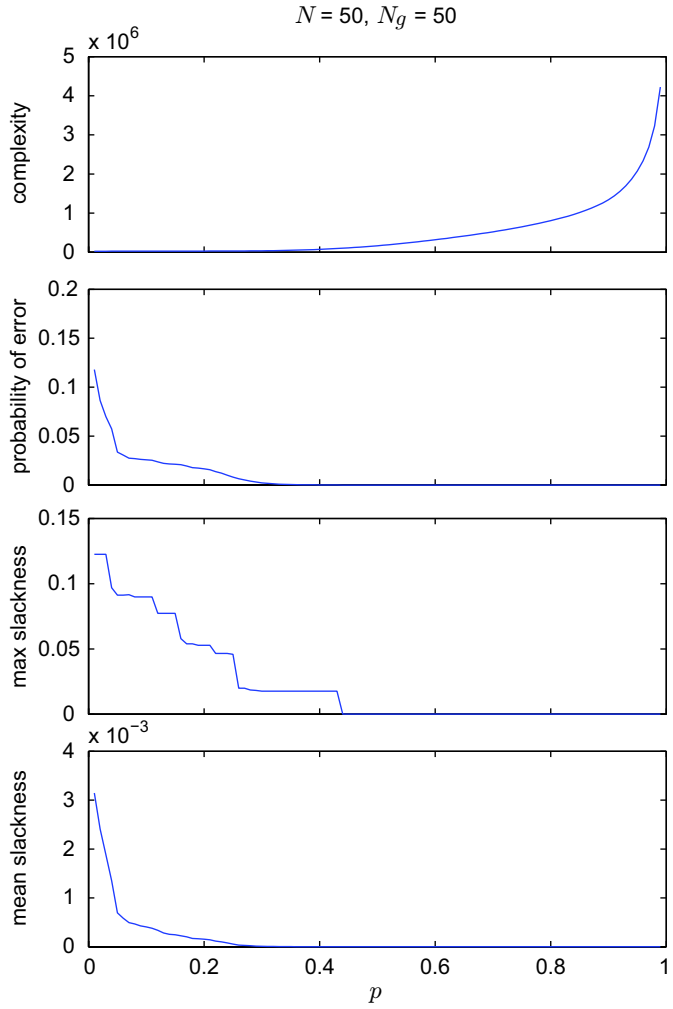


Figure 4.5 – *Real probability of error v.s. probabilistic model parameter. Comparing the value of the parameter p with the real probability error (middle up) and its influence on the complexity (up). The maximal slackness (middle down) and the mean slackness (down) are also presented. The algorithm uses a global probabilistic bound; the signals have 2000 samples, the database contains 10000 signals. The results are the mean of 20000 experiments.*

scale with a resolution of one third of octave. Due to isotropy, rotation is obviously useless for this kind of atoms.

An atom is uniquely defined by the set of parameters defining the generating function, the translation, the scaling and the rotation. Using images of size 128×128, the dictionary is made of 40271872 atoms, which makes it difficult to store the Gram matrix. However, due to their particular analytical form, it is possible to compute at low cost any entry of the gram matrix knowing the parameters of the atoms without having to create them.

Figure 4.7 presents the last steps of the execution of the algorithm when one or more candidates are eliminated. The candidates are shown using the number of atoms at disposal at the current step. The cardinality of the set of face images is 300; at step $K = 17$, corresponding to the second row of the figure, 9 candidates are remaining. The images

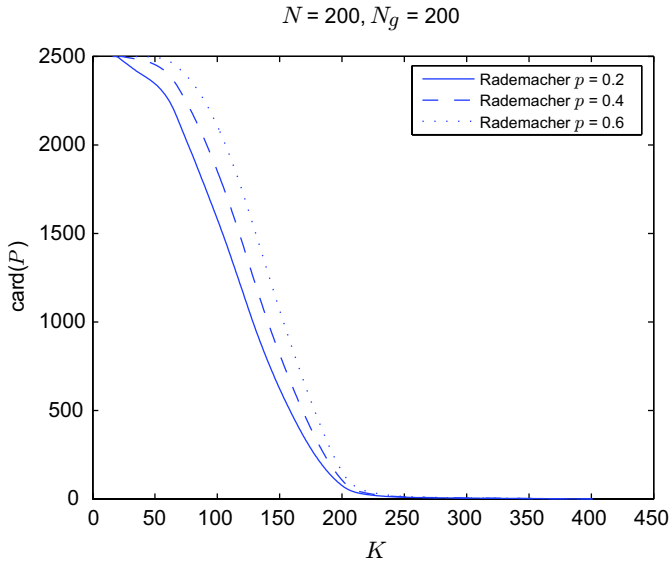


Figure 4.6 – *Number of candidates during the execution of the algorithm for images approximated with wavelets.*

representing the same person are likely to have a strong correlation with the reference image and are massively present in the remaining candidates.

The same experiment has been done with wavelet representations of images from the COIL-100 database [58]. A simple pretreatment consisting in normalizing the energy of the images has been done. The database contained 1500 images chosen randomly and all the images of size 128x128 were approximated using 1000 terms. Figure 4.9 presents the evolution of the cardinality of the set of potential candidates during the first 85 steps. The following steps are presented by Figure 4.8. The first row presents the query image and the images present in P after 85 steps reconstructed using 1000 wavelets. In the next rows, the images are reconstructed using only the wavelets that have been taken into account by the algorithm at this step. The algorithm is efficient in eliminating signals that are not from the *good* class. The four last rows contain the same object and as they are very similar, the algorithm need many steps to identify the best one. However, as the cardinality of the set of potential candidates is very low, the complexity is low too.

4.6 Discussion

The sparse structure of compressible signals offers a rather straightforward way to reduce the dimensionality of complex signals. We have exploited this structure to recursively localize those elements of a large set of signals that are closest to a fixed query. Our technique requires fundamental inputs. First, the coefficients of the expansion of each signal in the database must be stored and easily accessible. Note this is not a particularly stringent requirement since it is very likely that one would store compressed signals, using precisely the sparsity of their representation over a given basis or dictionary. Our technique is then

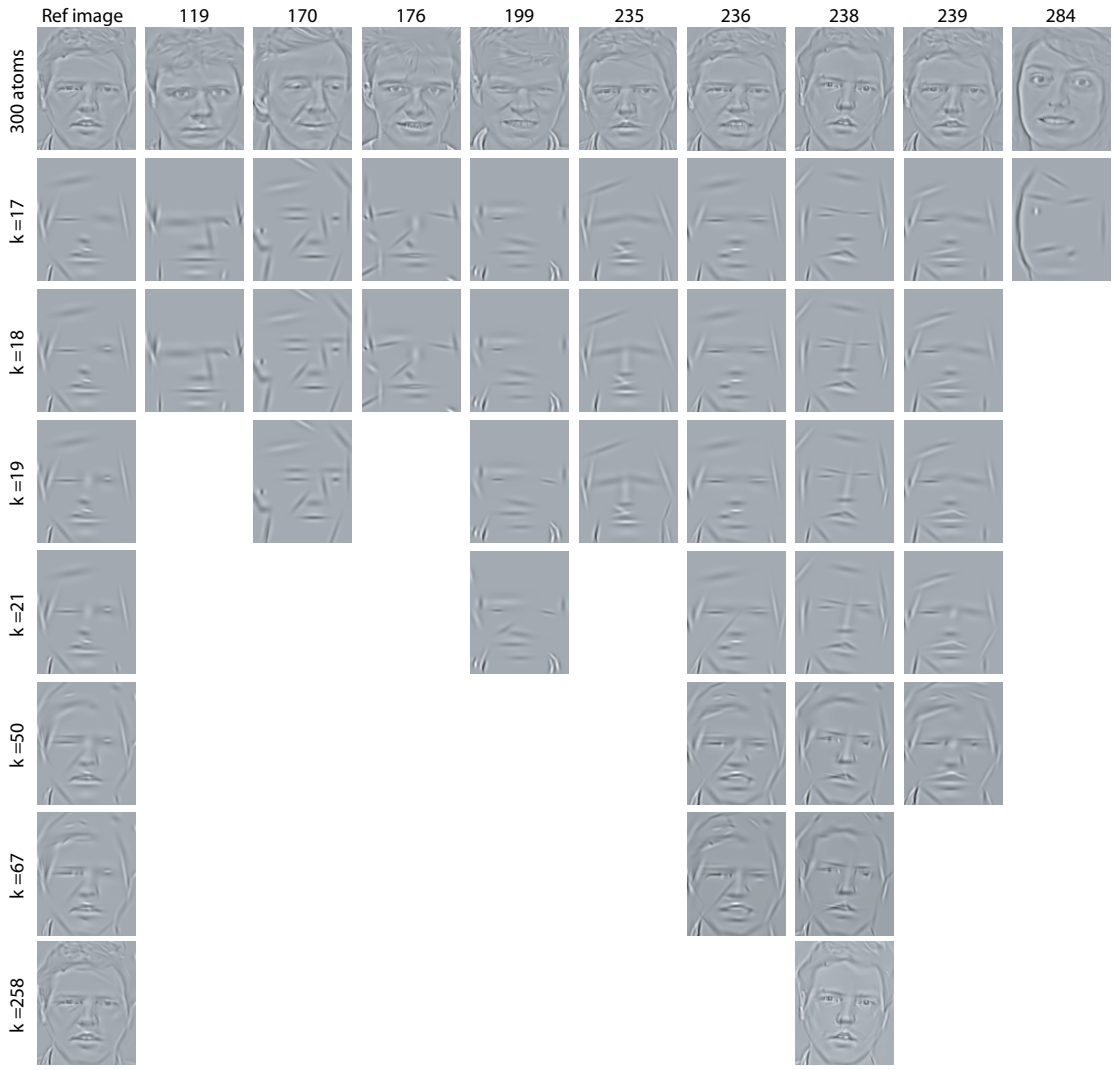


Figure 4.7 – *Last steps of the execution of the algorithm. The column most left presents the reference images whilst the other contain the candidates. The first row contains all fully recreated images whilst in the other rows, the images are reconstructed according to the number of atoms the algorithm takes into account.*

able to work on the compressed signals, in the sense that one doesn't have to reconstruct them for processing. Second, the gram matrix of the dictionary used to express signals must be stored or computed, too. If this not a problem when the dictionary is an orthogonal basis, it could be a severe limitation in the case of a general redundant dictionary since the gram matrix is a priori large and without particular structure. However, the gram matrix entries of many dictionaries used in practice can be computed in a fast way.

We showed that it is possible to maintain deterministic or probabilistic bounds on the true distance between the query and the tested signals. Clearly though, probabilistic bounds are much more favorable than our worst case deterministic bounds. Indeed, we presented clear experimental evidence showing the ability of the algorithm to eliminate non suitable candidates at early stages.

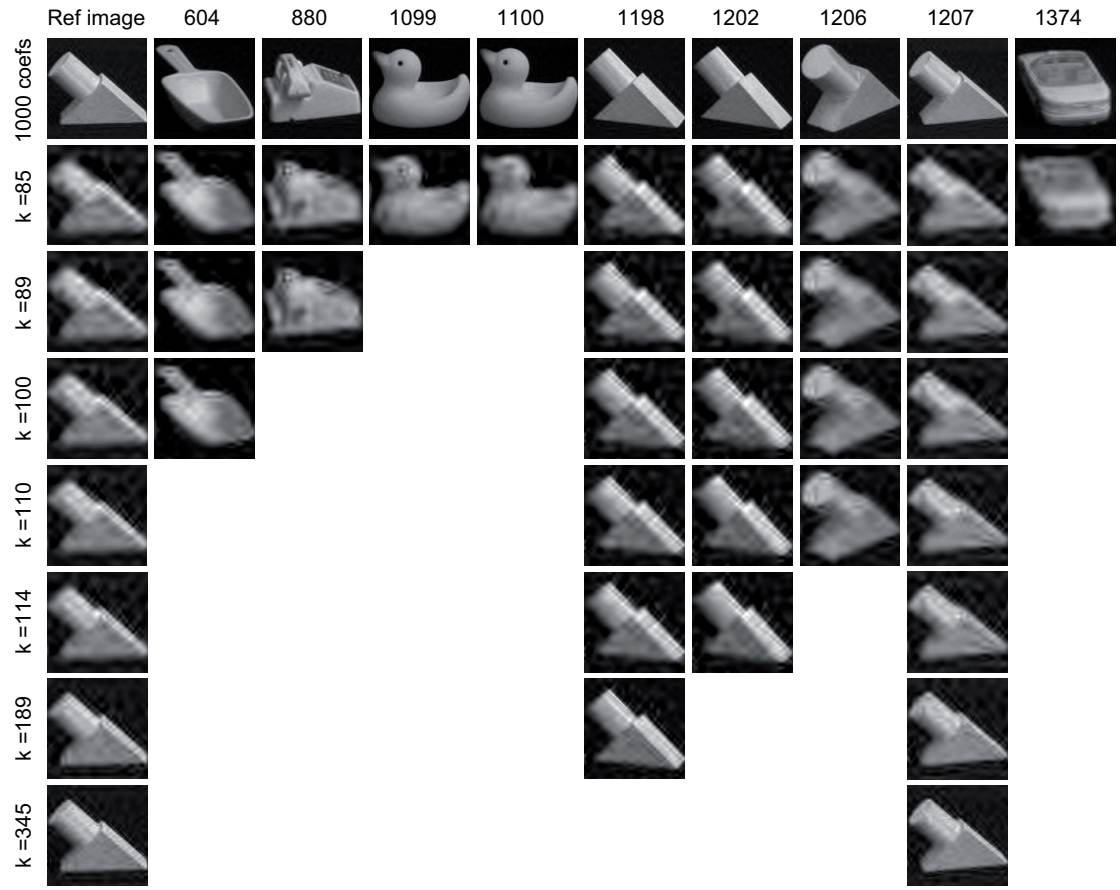


Figure 4.8 – *Last steps of the execution of the algorithm using wavelets approximation. The column most left presents the reference images whilst the other contain the candidates. The first row contains all fully recreated images (1000 wavelets) whilst in the other rows, the images are reconstructed according to the number of wavelets the algorithm takes into account.*

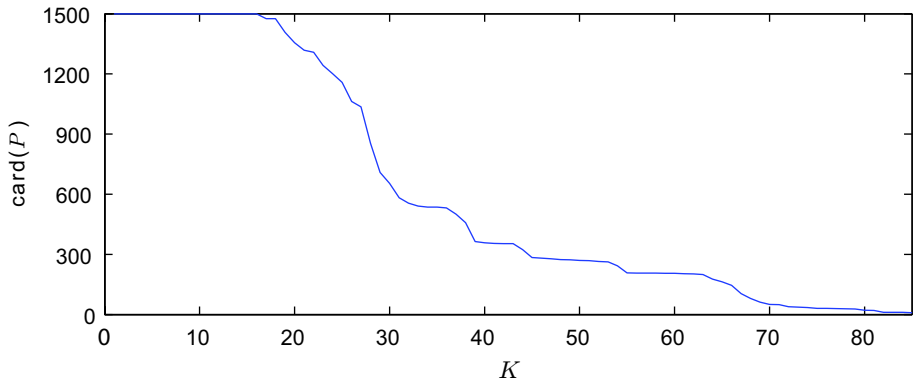


Figure 4.9 – *Evolution of the cardinality of the set of potential candidates before reaching the state shown in Figure 4.8.*

5 Conclusions

5.1 Achievements

Redundant expansions permit to express the information contained in signals in a meaningful way. However, almost all algorithms in this field are prohibitive in terms of computations or memory requirements. On the other hand, sparse approximations provide flexible representations that can lead to efficient treatments of the information and smart solutions to solve typical signal processing problems. Our main objective is to provide algorithmic solutions that could help propagating the sparse approximations techniques to real world applications.

Chapter 2 presented an algorithm, called Tree-Based Pursuit, which reduces the complexity of finding sparse representations with limited impact on the approximation rate. Additionally, thanks to the clustering technique used to create the dictionary data structure, this algorithm is able to recover coarse structures of the signals. This chapter studied highly redundant dictionaries which are often used but were rarely studied; different researchers obtained interesting results for incoherent dictionaries. We demonstrated that from a molecular point of view, their results also apply to highly redundant dictionaries.

The dictionary learning algorithm presented in Chapter 3 is less complex than other state of the art methods. This difference makes it possible to create bigger atoms and more populated dictionaries at a lower cost. Additionally, the shift invariance property allowed for finding waveforms that were not learnt with other techniques e.g. curved edge detectors. We believe that creation of dictionary using data mining techniques will eventually lead to efficient applications. In some situations, we may even expect to learn more about the underlying physic by finding good families of generating functions.

The last topic, treated in Chapter 4, illustrates that sparse approximations may be ex-

ploited to provide efficient signal processing tasks. The sparse structure of the coefficients was used to reduce the dimensionality of complex signals. The proposed solution has only few requirements. The most problematic one is to have a fast method to get entries of the Gram matrix. Accessing the sparse coefficients is generally not a problem as huge databases generally store compressed versions of the data. The proposed solution demonstrates that sparsity may be exploited in an intuitive way to solve complex tasks. Additionally, compressible signals are very common signals as this class contains natural images, sounds and music for example.

5.2 Future research directions

Combine Tree-Based Pursuit and MPTK. Recently, an efficient implementation of Matching Pursuit, called the Matching Pursuit Toolkit (MPTK) has been released. The authors worked on the practical bottlenecks of Matching Pursuit. They showed that they were not always where most researchers thought they would be. Finding the best node at the first level of the tree is equivalent to Matching Pursuit. Thus, the improvement MPTK brings to Matching Pursuit are directly applicable to Tree-Based Pursuit.

Further exploit the tree structure. Each atom in the dictionary correspond to a path in the tree. This equivalence permits to think about different possible applications. Sparse approximations allows for design efficient and low bit rate coder. Generally, these algorithms sort the selected atoms in decreasing order of magnitude of the associated projections. Adapted quantization schemes are used for the projections. In this case, the index of the atoms have to be encoded at a cost roughly equal to $\lceil \log_2 |\mathcal{D}| \rceil$. In some cases, the projections of the molecules on the path to the atom have comparable magnitude. Thus, from a coding point of view, they carry enough information to be used instead of the true atoms. It would be possible to make a coder that also uses molecules. When coding an atom by its path in the tree, we could hope to have a gain in compression.

MoTIF with other invariant transforms. Our learning algorithm finds translation invariant features. Obviously, MoTIF could be adapted to work under other invariant transforms at the condition that they admit a well defined adjoint. In a sense, this topic was already treated by Lesage in his thesis. He used this principle and the K-SVD structure from Aharon to define a new learning algorithm.

Multichannel MoTIF. Monaci used MoTIF to find multi-modal generating functions whilst keeping the important features present in the signals synchronized. We presented his alternate learning procedure that corresponds to localize in one modality and to learn a new generating function in the other one. This solution works remarkably well for two channels. Can it be generalized for more channels? How should the localize/learn paradigm be adapted in this situation? These questions remain open

and are in our opinion of great interest. Data mining on complex synchronized data represents a big challenge and should gain in interest in the future.

Nearest Neighbor with random dictionaries. In view of the recent results in compressed sensing, one may wonder wether it would be possible to avoid computing sparse approximations over a fixed dictionary since most of the information of compressible signals can be captured by random projections [10]. Exploring the possibility of working solely with random projections may be interesting future research direction.

Nearest Neighbor and kernel distances. The proposed algorithm computes distances between signals in function of the scalar products of their constituting atoms. Different applications use more complicated distances than the simple euclidian distance. Some of these distances are solely function of the scalar products between the signal. Thus, our solution may be adapted to iteratively compute these distances based on the inter-atoms scalar products. Finding the appropriate bounds may be more complicated. Some classification algorithm as Support Vector Machines rely heavily on selecting particular data points based on their scalar products with reference vectors. These techniques could potentially benefit from the fast rejection provided by the proposed Nearest Neighbor algorithm.

Bibliography

[1] M. Aharon, M. Elad, and A. M. Bruckstein. K-SVD and its non-negative variant for dictionary design. In *Proceedings of the SPIE conference wavelets*, volume 5914, July 2005.

[2] M. Aharon, M. Elad, and A. M. Bruckstein. K-SVD: Design of dictionaries for sparse representations. In *Proceedings of SPARS'05*, 2005.

[3] M. Aharon, M. Elad, and A. M. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, November 2006.

[4] M. Aharon, M. Elad, and A. M. Bruckstein. On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them. *Linear Algebra and its Applications*, 416:4867, 2006.

[5] Applied and Computational Mathematics - California Institute of Technology (Caltech). $\ell - 1$ *magic*. http://www.acm.caltech.edu/l1magic/.

[6] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu. An optitmal algorithm for approximate nearest neighbor searching in fixed dimensions. In *Proceedings of 5th ACM-SIAM SODA*, 1994.

[7] A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[8] T. Blumensath and M. Davies. Sparse and shift-invariant representations of music. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):50– 57, January 2006.

[9] D. M. Bradley and R. C. C. Gupta. On the distribution of the sum of n non-identically distributed uniform random variables. *Annals of the Institute of Statistical Mathematics*, 54(3):689–700, 2002.

[10] E. J. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59:1207–1223, 2005.

[11] S. S. Chen, D. L. Donoho, and M. A Saunders. Atomic decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

[12] K.-P. Cheung and Y.-H. Chan. A fast two-stage algorithm for realizing Matching Mursuit. In *Proceedings of IEEE International Conference on Image Processing (ICIP'01)*, volume 2, pages 431–434, October 2001.

[13] Y. T. Chou, W. L. Hwang, and C. L. Huang. Gain-shape optimized dictionary for Matching Pursuit video coding. *Signal Processing*, 83:1937–1943, September 2003.

[14] K. Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17:830–847, 1988.

[15] K. Clarkson. An algorithm for approximate closest-point queries. In *Proceedings of 10th ACM Symposium on Computational Geometry*, 1994.

[16] S. F. Cotter and B. D. Rao. Application of tree-based searches to Matching Pursuit. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'01)*, volume 6, pages 3933–3936, May 2001.

[17] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[18] I. Daubechies. Time-frequency localization operators: a geometric phase space approach. *IEEE Transactions on Information Theory*, 34:605–612, 1988.

[19] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Journal of Constructive Approximation*, 13:57–98, 1997.

[20] C. De Vleeschouwer and B. Macq. Subband dictionaries for low-cost matching pursuits of video residues. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(7):984–993, October 1999.

[21] I. S. Dhillon, E. M. Marcotte, and U. Roshan. Diametrical clustering for identifying anti-correlated gene clusters. *Bioinformatics*, 19(13):1612–1619, 2003.

[22] D. Dobkin and R. Lipton. Multidimensional search problems. *SIAM Jouranl of Computing*, 5:181–186, 1976.

[23] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100:2197–2202, March 2003.

[24] D. L. Donoho, M. Vetterli, R. A. DeVore, and I. Daubechies. Data compression and harmonic analysis. *IEEE Transactions on Information Theory*, 44:391–432, August 1998.

[25] M. Elad and A. M. Bruckstein. A generalized uncertainty principle and sparse representations in pairs of bases. *IEEE Transactions on Information Theory*, 48(9):2558–2567, September 2002.

[26] K. Engan, S. O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'99)*, volume 5, pages 2443–2446, 1999.

[27] K. Engan, S. O. Aase, and J.H. Husoy. Multi-frame compression: theory and design. *Signal Processing*, 80(10):2121–2140, October 2000.

[28] K. Engan, K. Skretting, and Husoy J. H. Family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation. *Digital Signal Processing*, 17(1):32–49, January 2007.

[29] R. M. Figueras i Ventura and P. Vandergheynst. *Sparse image approximation with application to flexible image coding.* PhD thesis, EPFL, Lausanne, 2005.

[30] R. M. Figueras i Ventura, P. Vandergheynst, and P. Frossard. Low rate and flexible image coding with redundant representations. *IEEE Transactions on Image Processing*, 15(3):726–739, March 2006.

[31] P. Frossard and P. Vandergheynst. Redundancy in non-orthogonal transforms. In *IEEE International Symposium on Information Theory*, page 196, June 2001.

[32] M. Goodwin and M. Vetterli. Matching Pursuit and atomic signal models based on recursive filter banks. *IEEE Transactions on Signal Processing*, 47(7):1890–1902, July 1999.

[33] I. F Gorodnitsky and B. D. Rao. Sparse signal reconstruction from limited data using FOCUSS: are-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing*, 45(3):600–616, March 1997.

[34] R. Gribonval. Fast matching pursuit with a multiscale dictionary of gaussian chirps. *IEEE Transactions on Signal Processing*, 49(5):994–1001, May 2001.

[35] R. Gribonval and S. Krstulovic. MPTK: Matching Pursuit made tractable. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP'06)*, May 2006.

[36] R. Gribonval and M. Nielsen. Sparse representations in unions of bases. *IEEE Transactions on Information Theory*, 49(12):3320–3325, December 2003.

[37] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[38] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Prodeedings of the 29th Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

[39] D. Jiang, C. Tang, and A Zhang. Cluster analysis for gene expression data: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, November 2004.

[40] P. Jost, P. Frossard, and P. Vandergheynst. Redundant image representations in security applications. In *Proceedings of IEEE International Conference on Image Processing (ICIP'04)*, volume 4, pages 2151–2154, October 2004.

[41] J. Karvanen and A. Cichocki. Measuring sparseness of noisy signals. In *Proceedings of 4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 125–130, April 2003.

[42] J. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Prodeedings of the 28th Symposium on Theory of Computing (STOC)*, pages 599–608, 1997.

[43] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. W. Lee, and T.J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Computation*, 15:349–396, 2003.

[44] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal of Computing*, 30:457–474, 2000.

[45] M. Ledoux and M. Talagrand. *Probability in Banach spaces : isoperimetry and processes*. Springer-Verlag, 1991.

[46] T. W. Lee, M. S. Lewicki, M. Girolami, and T. J. Sejnowski. Blind source separation of more sources than mixtures using overcomplete representations. *IEEE Signal Processing Letters*, 6(4):87–90, April 1999.

[47] S. Lesage. *Apprentissage de dictionnaires structurés pour la modélisation parcimonieuse des signaux multicanaux*. PhD thesis, Université de Rennes 1, 2007.

[48] M.S. Lewicki and B. A. Olshausen. A probabilistic framework for the adaptation and comparison of image codes. *Journal of the Optical Society of America*, 16(7):1587–1601, 1999.

[49] M.S. Lewicki and T.J. Sejnowski. Learning overcomplete representations. *Neural computation*, 12(2):337–365, 2000.

[50] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[51] S. Mallat and Z. Zhang. Matching Pursuit with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, December 1993.

[52] S. Marcel, P. Jost, P. Vandergheynst, and J. Thiran. Face authentication using client-specific Matching Pursuit. Technical report, EPFL, 2004.

[53] S. K. Mitra. On the probability distribution of the sum of uniformly distributed random variables. *SIAM Journal on Applied Mathematics*, 20(2):195, 1971.

[54] G. Monaci. *On the modelling of Multi-modal data using redundant dictionaries*. PhD thesis, EPFL, 2007.

[55] G. Monaci, P. Jost, P. Vandergheynst, B. Mailhe, S. Lesage, and R. Gribonval. Learning multi-modal dictionaries: Application to audiovisual data. In *Proceedings of International Workshop on Multimedia Content Representation, Classification and Security*, pages 538–545, 2006.

[56] G. Monaci, P. Jost, P. Vandergheynst, B. Mailhe, S. Lesage, and R. Gribonval. Learning multi-modal dictionaries. *IEEE Transactions on Image Processing*, 2007.

[57] R. Neff and A. Zakhor. Matching Pursuit video coding .i. dictionary approximation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(1):13–26, January 2002.

[58] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). Technical report, CUCS-006-96, February 1996.

[59] B. A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[60] B. A. Olshausen and D.J. Field. Natural image statistics and efficient coding. *Network: Computation in Neural Systems*, 7:333–339, 1996.

[61] B. A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

[62] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal Matching Pursuit: Recursive function approximations to wavelet decomposition. In *Proceeding of the 27th Annual Asilomar Conference on Signals Systems and Computers*, 1993.

[63] L. Peotta, L. Granai, and P. Vandergheynst. Very low bit rate image coding using redundant dictionaries. In *Proceedings of the SPIE, Wavelets: Applications in Signal and Image Processing*, volume 5207 of *Lecture Notes in Computer Science*, pages 228–239. SPIE, 2003.

[64] L. Peotta, P. Jost, P. Vandergheynst, and P. Frossard. Sparse approximation with block incoherent dictionaries. TR - ITS 2003.007, EPFL, 1015 Ecublens, December 2003.

[65] M. D. Plumbley. Recovery of sparse representations by Polytope Faces Pursuit. In *Proceedings of the 6th International Conference on Independent Component Analysis and Blind Source Separation (ICA 2006)*, LNCS 3889, pages 206–213, Charleston, SC, USA, March 2006. Springer Verlag, Berlin.

[66] D.W. Redmill, D. R. Bull, and P. Czerepinki. Video coding using a fast non-separable Matching Pursuits algorithm. In *Proceedings of IEEE International Conference on Image Processing (ICIP'98)*, volume 1, pages 769–773, October 1998.

[67] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):660–674, May/June 1991.

[68] F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification. In *IEEE Workshop on Applications of Computer Vision*, Sarasota (Florida), December 1994.

[69] P. Schmid-Saugeon and A. Zakhor. Dictionary design for Matching Pursuit and application to motion-compensated video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6):880 – 886, June 2004.

[70] Signal Processing Institute - Swiss Federal Institute of Technology (EPFL). *JJ2000*. http://jj2000.epfl.ch/.

[71] E. C. Smith and M. S. Lewicki. Efficient coding of time-relative structure using spikes. *Neural Computation*, 17(1):19–45, 2005.

[72] E. C. Smith and M. S. Lewicki. Learning efficient auditory codes using spikes predicts cochlear filters. In *Advances in Neural Information Processing Systems*, volume 17, pages 1289–1296, 2005.

[73] E. C. Smith and M. S. Lewicki. Efficient auditory coding. *Nature*, 439:978–982, February 2006.

[74] J.-L. Starck, E. Candès, and D. L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 11(6):131141, 2002.

[75] J.-L. Starck, M. Elad, and D. L. Donoho. Redundant multiscale transforms and their application for morphological component analysis. *Advances in Imaging and Electron Physics*, 132:287348, 2004.

[76] V. N. Temlyakov. Weak greedy algorithms. *Advances in Computational Mathematics*, 12(2-3):213–227, February 2000.

[77] V. N. Temlyakov. Nonlinear methods of approximation. *Foundations of Computational Mathematics*, 3(1):33–107, January 2003.

[78] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, October 2004.

[79] J. A. Tropp. Just relax: Convex programming methods for subset selection and sparse approximation. ICES Report 04-04, The University of Texas at Austin, February 2004.

[80] J. A. Tropp. *Topics in Sparse Approximation*. Computational and applied mathematics, UT-Austin, August 2004.

[81] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, June 1991.

Curriculum vitæ

Name:	Philippe Jost
Citizenship:	Swiss
Birthdate:	December 2, 1978
Birthplace:	Vevey, Switzerland
Marital status:	Single

Contact information

Address:	Quai Perdonnet 14
	1800 Vevey, Switzerland
Phone:	+41 21 693 26 57
Fax:	+41 21 693 76 00
Email:	philippe.jost@epfl.ch
Web page:	http://lts2www.epfl.ch/∼jost

Work experience

- ***April 2003 – present***: Research assistant at the Signal Processing Institute, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland
  - PhD Thesis research in the field of sparse approximation using redundant dictionaries.
  - Supervision of master thesis.
  - Responsible of exercises and laboratory activities for the Digital Signal Processing and for postgraduate course: Advanced Digital Image Processing and Analysis.
  - Webmaster (`http://lts2www.epfl.ch` and `http://itswww.epfl.ch`).
- ***Summer 2003***: Master Thesis at Visiowave's Research department
  - Dynamic Region and Block-Based Motion Estimation for Video Compression.

- ***Summer 2001***: Intership at Visiowave's Research department
  - Motion-Based Frame Interpolation.
  - Multi-scale Block-Based Motion Estimation.
- ***1999 – present***: Screenager.net
  - Webhosting.
  - Development of dynamic websites.
- ***Summer 1999***: Internship at IBM - Geneva - Switzerland
  - Internal customer support.
  - In charge of backup servers.
  - Software Installation.
- ***Summer 1997***: Internship at Nestlé - Vevey - Switzerland
- ***Summer 1996***: Working for the Youth Hostel of Montreux - Switzerland

Education

- ***October 2003 – present***: *Ph. D. student* in sparse approximation. Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland.
- ***October 2003 – present***: *Doctoral School* in Computer, Communication and Information Science. Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland.
- ***1997 – 2002***: *M.s.* in Communication Systems, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland.

Scientific Activities and Seminars

- *Reviewer* for IEEE Transactions on Signal Processing.
- ***July 2005***: *Visiting Scientist* Institut de Recherche en Informatique et Systèmes aléatoires (IRISA) , Rennes, France.
- ***June 2005***: *Visiting Scientist* Numerical Harmonic Analysis Group (NuHAG), University of Vienna, Austria.
- ***June 2005***: Special semester on Modern Methods in Time-Frequency Analysis, University of Vienna, Austria.

Skills

Languages

French:	mother tongue.
Swiss German:	mother tongue.
English:	fluent oral and written.
German:	fluent oral and written.

Computer literacy

Operating systems:	UNIX, Linux, Windows
Programming Languages:	JAVA, C, C++, SQL, PHP, XML, UM
Other:	Matlab, Mathematica, Maple, Office, HTML, LaTeX, Apache

Extra-curricular activities

- *Sports* : Golf (HCP 7), Snowboard, Tennis
- *Associative* : Zofingue: President winter semester 2001/02 and winter semester 2005/06
- *Reading*, *Music*, *Arts*

Personal publications

Journal papers

- P. Jost, P. Vandergheynst, *On finding nearest neighbors in a set of compressible signals*, submitted to IEEE Transactions on Signal Processing, 2007.
- G. Monaci, P. Jost, P. Vandergheynst, B. Mailhe, S. Lesage and R. Gribonval, *Learning Multi-Modal Dictionaries*, submitted to IEEE Transactions on Image Processing, 2007.
- P. Jost, P. Vandergheynst, P. Frossard, *Tree-Based Pursuit: Algorithm and Properties*, IEEE Transactions on Signal Processing, Vol. 54, Nr. 12, pp. 4685-4697, 2006.

Conference papers

- G. Monaci, P. Jost, P. Vandergheynst, B. Mailhe, S. Lesage, R. Gribonval, *Learning Multi-Modal Dictionaries: Application to Audiovisual Data*, Workshop on Multimedia Content Representation, Classification and Security, in Springer-Verlag LNCS series, Vol. 4105, pp. 538-545, 2006.
- P. Jost, S. Lesage, P. Vandergheynst, R. Gribonval, *MoTIF: An Efficient Algorithm for Learning Translation Invariant Dictionaries*, International Conference on Acoustics, Speech, and Signal Processing (IEEE ICASSP'06), 2006.
- P. Jost, S. Lesage, P. Vandergheynst, R. Gribonval, *Learning Redundant Dictionaries with Translation Invariance Property: the MoTIF Algorithm*, Structure et Parcimonie pour la Représentation Adaptative de Signaux (SPARS'05), 2005.
- G. Monaci, P. Jost, P. Vandergheynst, *Image compression with learnt tree-structured dictionaries*, International Workshop on Multimedia Signal Processing (IEEE MMSP'04), 2004.
- P. Jost, P. Frossard, P. Vandergheynst, *Redundant Image Representations in Security Applications*, International Conference on Image Processing (IEEE ICIP'04), 2004.

Technical reports

- P. Jost, P. Vandergheynst, P. Frossard, *Tree-Based Pursuit: Algorithm and Properties*, Technical Report, 2005.
- S. Marcel, P. Jost, P. Vandergheynst, J. Thiran, *Face Authentication using Client-specific Matching Pursuit*, Technical Report, 2004.
- P. Jost, P. Vandergheynst, P. Frossard, *Tree-Based Pursuit, Technical Report*, 2004.
- L. Peotta, P. Jost, P. Vandergheynst, P. Frossard, *Sparse Approximation with Block Incoherent Dictionaries*, Technical Report, 2003.

Master Thesis

- P. Jost, J. Reichel, F. Ziliani, P. Vandergheynst, *Dynamic Region and Block-Based Motion Estimation for Video Compression*, Ecole Polytechnique Fédérale de Lausanne (EPFL), 2002.

Patents

- P. Jost, P. Vandergheynst, P. Frossard, *A system for very low rate face image compression and authentication.*